TECHNICAL REPORT ARBRL-TR-02206

# A NUMERICAL METHOD TO INTEGRATE STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

T. P. Coffee
J. M. Heimerl
M. D. Kregel

January 1980

US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND
BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

Secondary distribution of this report by originating or sponsoring activity is prohibited.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22151.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

*The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.*

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: TECHNICAL REPORT ARBRL-TR-02206
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: A NUMERICAL METHOD TO INTEGRATE STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

5. TYPE OF REPORT & PERIOD COVERED: BRL Technical Report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s):
T. P. Coffee
J. M. Heimerl
M. D. Kregel

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
USA Armament Research and Development Command
USA Ballistic Research Laboratory
ATTN: DRDAR-BLP
Aberdeen Proving Ground, MD 21005

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
RDT&E Project #1L161102AH43

11. CONTROLLING OFFICE NAME AND ADDRESS:
USA Armament Research and Development Command
USA Ballistic Research Laboratory
ATTN: DRDAR-BL
Aberdeen Proving Ground, MD 21005

12. REPORT DATE: JANUARY 1980

13. NUMBER OF PAGES: 54

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:

15. SECURITY CLASS. *(of this report)*: Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*:
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:

18. SUPPLEMENTARY NOTES:

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:
Numerical Integration
Stiff Ordinary Differential Equations
Jacobian Matrix
Predictor-Corrector Methods

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)* srf

A method is described for the efficient integration of stiff systems of ordinary differential equations. The method, based on a predictor-corrector formulation, uses the Jacobian in a non-standard fashion. The resulting program is compared with EPISODE, a standard stiff integrator, for a number of systems of ordinary differential equations. The results show that the procedure is usually competitive with EPISODE and is more efficient for some problems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

TABLE OF CONTENTS

# I. INTRODUCTION

The K-integrator for stiff ordinary differential equations was developed in the late 1960's by one of us (MDK), and has been repeatedly improved since that time[1-3]. It is intended to solve systems of the form

$$\dot{Y}(t) = G\,(Y(t),t), \quad t\varepsilon[a,b], \tag{1}$$

$$Y(a) = Y_0 \quad ,$$

where $\dot{Y}(t) = dY(t)/dt$,

$$Y(t) = [Y^1(t),\ Y^2(t),\ \ldots\ ,\ Y^N(t)]^T\ ,$$

and $$G(u,t) = [G^1(u,t),\ G^2(u,t),\ \ldots\ ,\ G^N(u,t)]^T.$$

The concept of stiffness is difficult to define formally. However, it can be described in terms of the Jacobian matrix J(u,t), where the element in row i and column j is

$$J^{ij}(u,t) = \partial G^i(u,t)/\partial y^j. \tag{2}$$

A stiff system will have one or more eigenvalues in the Jacobian whose real parts are negative and large in modulus. As a result the corresponding components in the solution will decay very rapidly in comparison to the other terms present.

---

[1] *M. D. Kregel and E. L. Lortie, "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations", BRL Report No. 1733, July 1974 (ADA #A003855).*

[2] *M. D. Kregel and J. M. Heimerl, "Comments on the Solutions of Coupled Stiff Differential Equations", BRL Memorandum Report No. 2769, July 1977; or Proceedings of the 1977 Army Numerical Analysis and Computers Conference, ARO Report 77-3, November 1977, pp. 553-563 (ADA #A043122).*

[3] *T. P. Coffee, J. M. Heimerl, and M. D. Kregel, "A Numerical Method for Large Stiff Systems of Ordinary Equations", Transactions of the 24th Conference of Army Mathematicians, ARO Report 79-1, January 1979. pp. 249-257.*

It will be useful to rewrite the system (1) using the diagonal Jacobian elements. We define the diagonal matrix $R(Y(t),t)$ and the vector $F(Y(t),t)$ with components,

$$R^i(Y(t),t) = -J^{ii}(Y(t),t), \text{ and} \tag{3}$$

$$F^i(Y(t),t) = \dot{Y}^i(t) + R^i(Y(t),t)Y^i(t).$$

Then equation (1) becomes

$$\dot{Y}(t) = F(Y(t),t) - R(Y(t),t)Y(t). \tag{4}$$

The term $R^i$ measures the sensitivity of $\dot{Y}^i$ with respect to changes in $Y^i$.

The system is solved at a set of discrete points in time. Thus, step sizes, $h_i$, and times, $t_n = a + \sum_{i=1}^{n} h_i$, are introduced. Approximations $y_n$ to $Y(t_n)$ are then produced for n=1,2, ... by the integrator.

These approximations will be found using multistep formulas of the form $y_n = \sum_{i=1}^{q} \alpha_i y_{n-i} + h_n \sum_{i=0}^{q} \beta_i \dot{y}_{n-i}$. (5)

The truncation error will depend on the order of the formula. For stiff equations, there is also the problem of stability. That is, a small error in one step may propagate and grow in subsequent steps. Standard non-stiff methods will be restricted to very small step sizes to preserve stability. Dahlquist[4] has shown that to maintain reasonable stability for stiff systems, the formulas must be implicit, that is, $\beta_0 \neq 0$.

Most stiff multi-step methods use a modified Newton iteration to solve (5). First an explicit predictor $y_n^P$ is found. Then its time derivative and corresponding corrector are defined:

$$\dot{y}_n^P = F(y_n^P, t_n) - R(y_n^P, t_n)y_n^P \quad \text{and} \tag{6}$$

---

[4] *G. G. Dahlquist, AMS Symp. Appl. Math. 15, (1963), 147.*

$$y_n^C = \sum_{i=1}^{q} \alpha_i y_{n-i} + h_n \sum_{i=1}^{q} \beta_i \dot{y}_{n-1} + h_n \beta_o \dot{y}_n^P. \qquad (7)$$

Finally the vector $\Delta_n$ is defined such that

$$y_n = y_n^P + \Delta_n. \qquad (8)$$

An approximation $d_n$ to $\Delta_n$ can be found from the difference $(y_n^P - y_n^C)$, by expanding $\dot{y}_n$ in a series about the $y_n^P$ and truncating. This procedure leaves the following system of linear equations to be solved,

$$(y_n^P - y_n^C) = [h\beta_o J_n - I]d_n, \qquad (9)$$

where $J_n = J(y_n^P, t_n)$ and I is the identity matrix. The accuracy of the approximation to $y_n$ can be found by monitoring the size of the quantity $(y_n^{P2} - y_n^{C2})$, where

$$y_n^{P2} = y_n^P + d_n \qquad \text{and} \qquad (10)$$

$$y_n^{C2} = \sum_{i=1}^{q} \alpha_i y_{n-i} + h_n \sum_{i=1}^{q} \beta_i \dot{y}_{n-i} + h_n \beta_o \dot{y}_n^{P2}. \qquad (11)$$

If necessary, the Newton iteration can be repeated.

Solving the system of equations (9) requires approximately $N^3/3$ multiplications and divisions, where N is the number of equations. Thus, in general the larger N, the greater the computation time. Reduction of this time has been an objective of several algorithms including the one presented herein.

One approach to this problem is the well-known algorithm DIFSUB, by C. W. Gear[5]. This is a variable-order method, based on formulas of the form

---

[5] *C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, 1971.*

$$y_n = \sum_{i=1}^{q} \alpha_i y_{n-i} + h\,\beta_0\,\dot{y}_n, \quad 1 \le q \le 6. \tag{12}$$

The program automatically uses the higher order formulas if more accuracy is required. All the formulas are based on a fixed-step size h. To change the step size, the appropriate values for $y_{n-i}$ are found by interpolation.

For a given step, the matrix based on the Jacobian is found and inverted. The program reevaluates this matrix only when it fails to obtain convergence in three Newton iterations. One therefore does not have to invert a matrix every step.

More recently, variable-step formulas have been developed. Any sequence of step sizes $h_i$ can be used. The coefficients $\alpha_i$ and $\beta_i$ in (5) are variable and depend on the previous step sizes. In practice, such codes have been more stable.

One such code is EPISODE, developed by Byrne and Hindmarsh[6]. It uses a variable-order, variable-step formula. Like DIFSUB, it uses an aged Jacobian. An LU decomposition is computed instead of a matrix inversion.

The K-integrator uses a fixed-order, variable-step formula. Except for starting, it always uses a third-order formula. The lack of higher order formulas will not be important, provided only moderate accuracy is required.

The major innovation in the K-integrator is the method of solving the set of equations (9). Both DIFSUB and EPISODE approximate $(h\beta_o J_n - I)$ by the values of the matrix at an earlier time step $t_{n-i}$. The K-integrator evaluates the matrix at each time step; but, if possible, it is one of reduced size. Finding a suitable approximation to any of the $d_n^k$ permits the corresponding row and column to be eliminated from the matrix.

To illustrate, we write (9) in component form; i.e.,

$$y_n^{Pk} - y_n^{Ck} = h\beta_o \sum_{\substack{j=1 \\ j\neq k}}^{N} J_n^{kj}\, d_n^j + [h\beta_o J_n^{kk} - 1]\, d_n^k. \tag{13}$$

---

[6] *G. D. Byrne and A. C. Hindmarsh,* ACM Trans. Math. Software *(1975), 71.*

We attempt to approximate $d_n^k$ by

$$d_n^k = (y_n^{Pk} - y_n^{Ck})/(h\beta_o J_n^{kk} - 1). \qquad (14)$$

There are several cases where (14) is a reasonable approximation. First, if h is very small, such as at the beginning of the integration, so that $|h\beta_o \sum_{j\neq k} J_n^{kj} d_n^j| << |d_n^k|$, then the off-diagonal terms may be neglected. Second, even if h is large, the equation for $y_n^k$ may be weakly coupled to the other equations, i.e., $|J_n^{kk} d_n^k| >> |\sum_{j\neq k} J_n^{kj} d_n^j|$. Finally, at any value of h, $y_n^{Pk}$ may be a good approximation to $y_n^{Ck}$, and so $|y_n^{Pk} - y_n^{Ck}| \simeq 0 \simeq d_n^k$.

In general, the set of equations (13) is solved in two stages. First a set of indices, L, is determined for which (14) is valid, i.e.,

$$d_n^\ell = (y_n^{P\ell} - y_n^{C\ell})/(h\beta_o J_n^{\ell\ell} - 1), \qquad \ell\varepsilon L. \qquad (15)$$

Then, using these values, the system (13) becomes

$$y_n^{Pk} - y_n^{Ck} - h\beta_o \sum_{j\varepsilon L} J_n^{kj} d_n^j = h\beta_o \sum_{\substack{j\notin L \\ j\neq k}} J_n^{kj} d_n^j + [h\beta_o J_n^{kk} - 1] d_n^k, \qquad k \notin L. \qquad (16)$$

The reduced system (16) is then solved using an LU decomposition and back substitution.* Where the reduced system involves a small number of the original set of equations, for example when h is small, there will be a significant reduction in computation time.

The difficulty in using this algorithm lies in determining which set of equations can be solved with sufficient accuracy using (14). The version reported here monitors the error terms $|y_n^{P2k} - y_n^{C2k}|$ and uses them to determine the set of indices L for the next time step. The details are given in the next section.

---

**Earlier versions of the K-method (Refs. 1 and 2) included a Gauss-Seidel iteration procedure before the matrix factorization. The version reported here omits this iteration procedure.*

## II. THE NUMERICAL METHOD

The K-integrator is based on a fixed third order formula of the form

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} + h (\beta_0 \dot{y}_n + \beta_1 \dot{y}_{n-1} + \beta_2 \dot{y}_{n-2}) \quad (17)$$

It is a variable step formula. If h, h', and h'' are the last three step sizes, then

$$\beta_2 = 0.05$$

$$\beta_1 = 0.35$$

$$p = h'/h$$

$$q = (h' + h'')/h$$

$$\beta_0 = [(1+p)(1+q) - \beta_2 p(p-q) - \beta_1 pq]/(3+2p+2q+pq)$$

$$\alpha_3 = [1+p-\beta_2 p^2 - \beta_0(3+2p)]/[q^2(p-q)]$$

$$\alpha 2 = (\beta_2 + \beta_1 - \alpha_3 q + \beta_0 - 1)/p$$

$$\alpha_1 = 1 - \alpha_2 - \alpha_3 .$$

For constant step size, the stability can be described in terms of parameters $\alpha$ and D. A formula is said to be A ($\alpha$, D) stable, $\alpha \in (0, \frac{\pi}{2})$, if all numerical solutions to $\dot{Y} = -\lambda Y$ converge to zero as $n \to \infty$ with h fixed for all $|\arg(-\lambda h)| < \alpha$, $|\lambda| \neq 0$ and for all $\mathrm{Re}(h\lambda) \leq D$[7]. This combines the features of the A($\alpha$) - stability of Widlund[8] and the stiff stability of Gear[5]. For the above formula $\alpha = 81°$ and D = 0.4. The truncation error, neglecting higher order terms, is $-0.0966\, h^4 Y^{(4)}(t)$. If the step size is not held constant, the truncation error will vary slightly.

---

[7] *G. K. Gupta, Math. of Comp. 30 (1976), 417.*

[8] *O. B. Widlund, BIT 7 (1967), 65.*

In practice the K-integrator uses modified forms of equations (9) and (10) that are somewhat more convenient for chemical kinetics problems, where it has mainly been applied. The modified form of equation (9) is

$$y_n^P - y_n^C = A_n g_n \ , \tag{18}$$

where $A_n^{ij} = [h\beta_o J_n^{ij} - \delta_{ij}]y_n^{Pi}$ and

$$g_n^i = d_n^i / y_n^{Pi} \ .$$

In this notation equation (10) becomes

$$y_n^{P2i} = y_n^{Pi}(1 + g_n^i) \quad , \quad i = 1,2,\ldots,N. \tag{19}$$

The rationale for (18) and (19) is the following. For a given network or set of chemical reactions at a known temperature, the rate at which each reaction proceeds equals a constant multiplied by the concentration(s) of the chemical species involved. The modified Jacobian, with components $J_n^{ij}y_n^{Pi}$, can be generated by adding and subtracting rates, which is computationally easy to do. (To recover the Jacobian matrix written in terms of the rates alone, we divide by the $y_n^{Pi}$.) In the modified notation, the diagonal approximation (14) becomes

$$g_n^k = (y_n^{Pk} - y_n^{Ck})/A_n^{kk} \ . \tag{20}$$

In order to minimize the computation time, we find as many of the $g_n^k$ as possible using (20). As discussed in the Introduction, it is expedient to make $|y_n^{Pk} - y_n^{Ck}|$ as small as possible. To this end we have developed an unorthodox predictor. While not essential to the code, it does increase its efficiency.

The usual predictor is obtained from an explicit multistep formula. But difficulties may arise when $y_n^P$ is used in equation (8) to obtain $y_n^C$.

Since the system is stiff, the quantities $R^i(y_n^P, t_n)$ from equation (4) will often differ in absolute value by many orders of magnitude. In order to finish the integration in a reasonable time, h must become large enough so that some of the terms $h\beta_0 R^i$ become very large. This means small errors in $y_n^P$ can become greatly magnified in $y_n^C$.

This effect can be reduced by considering the form of equation (17) for the final value $y_n$. Using (4) this can be written as

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} + h \beta_1 \dot{y}_{n-1} \tag{21}$$

$$+ h \beta_2 \dot{y}_{n-2} + h \beta_0 F(y_n, t_n) - h \beta_0 R(y_n, t_n) y_n .$$

We shall derive a formula for $y_n^P$ that follows this general form.

First, we obtain predicted values for F and R, using the formulas

$$F_n^P = \gamma_1 F_{n-1} + \gamma_2 F_{n-2} + \gamma_3 F_{n-3} \tag{22}$$

$$R_n^P = \gamma_1 R_{n-1} + \gamma_2 R_{n-2} + \gamma_3 R_{n-3} .$$

The quantities $\gamma_1$, $\gamma_2$, and $\gamma_3$ are determined uniquely for any step sizes by imposing the condition that the formulas be second order. For the case of constant step size, the truncation errors are $h^3 F^{(3)}(y, t)$ and $h^3 R^{(3)}(y,t)$, neglecting higher order terms.

Substituting these predicted values into equation (21), we obtain

$$y_n^P = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} \tag{23}$$

$$+ h \beta_1 \dot{y}_{n-1} + h \beta_2 \dot{y}_{n-2} + h \beta_0 F_n^P - h \beta_0 R_n^P y_n^P .$$

This equation can be solved for $y_n^P$.

Neglecting higher order terms,

$$y_n^i - y_n^{Pi} = \frac{h^4 \beta_0 \left( F^{(3)i}(y, t) - R^{(3)i}(y, t)\, y_n^{Pi} \right)}{1 + h \beta_0 R^i(y_n, t_n)} \tag{24}$$

As $h \beta_0 R^i$ becomes large, the accuracy of $y_n^{Pi}$ tends to improve. Of course, errors still occur in $F_n^{Pi}$ and $R_n^{Pi}$; but, we no longer obtain large errors just because $h \beta_0 R^i$ is large. In fact, in the special case where F and R are constant, $y_n - y_n^P = 0$.

After the first few steps, a further heuristic modification is made in $y_n^P$. We define an error term for the predictor by the equation

$$E_n^{Pi} = (y_n^{P2i} - y_n^{Pi})/y_n^{Pi} \tag{25}$$

where $y_n^{P2}$ is the final accepted predicted value. Then we define a weighted average of these errors by

$$w_n^i = 0.5\, E_n^{Pi} + 0.5\, w_{n-1}^i, \tag{26}$$

Finally in the next step we define a modified predictor whose components are given by

$$y_{n+1}^{PMi} = y_{n+1}^{Pi} (1 + w_n^i). \tag{27}$$

This modified predictor is used in equation (7) to obtain the corrector $y_n^C$.

The purpose of this modification is to detect any systematic errors in the predictor. We assume that the accuracy of $y_n^P$ is similar to the accuracy of $y_{n-1}^P$. If the errors are not systematic, the fact that (26) is a weighted average will tend to minimize any errors introduced by the $w_n^i$.

The existence of systematic errors in an explicit predictor can be shown for the simple case of a single equation $\dot{Y} = -\lambda Y$, $Y(a) = Y_0$. Let $y_n^P$ be given by an explicit multistep formula $y_n^P = \sum_{i=1}^{q} \alpha_i y_{n-i} + h \sum_{i=1}^{q} \beta_i \dot{y}_{n-i}$, where h is constant. Assume the values at the previous steps are known exactly, that is, $y_{n-i} = Y(t_{n-i})$ and $\dot{y}_{n-i} = \dot{Y}(t_{n-i})$, $i=1,\ldots,q$. Then $(Y(t_n) - y_n^P)/y_n^P$ is a constant, independent of the values for both n and a. This can be seen simply by substitution into the above expression.

In general, we cannot prove that systematic errors occur. However, in practice this modification does lead to a noticeable improvement in the accuracy of the predictor.

The key step in the algorithm, the determination of the set of indices, L, for which the diagonal approximation (14) is valid, depends on a user supplied error tolerance, $\varepsilon$. (Error control is implemented on a per step basis.)

Since h at the start is made quite small, it is reasonable to solve for all the $g_n^i$ by (14). Thus, in this version of the code, all indices 1, 2, ..., N are put into L at the beginning of the integration. (So that the entire Jacobian does not have to be evaluated, the diagonal elements, $J_n^{kk}$ $y_n^k$, are computed in a separate subroutine.) Next we define a convergence error, $E_c^i$, for each component. If an absolute error criterion is used

$$E_c^i = |y_n^{C2i} - y_n^{P2i}| \; ; \tag{28a}$$

if a relative error criterion is used

$$E_c^i = 2\,|(y_n^{C2i} - y_n^{P2i})/(y_n^{C2i} + y_n^{P2i})| \; , \tag{28b}$$

where $y_n^{P2}$ and $y_n^{C2}$ are found from (17) and (11), respectively. In addition an overall root mean square (rms) convergence error, $E_c$, is defined by

$$E_c = \left( \sum_{i=1}^{N} (E_c^i)^2/N \right)^{1/2} . \qquad (29)$$

If $E_c < \varepsilon$ we accept $y_n^{C2}$ as our final value for $y_n$. Otherwise the Newton method is repeated. Also, if $E_c^i < \varepsilon/5$, the ith index remains in L. Otherwise the corresponding equation is solved using the analogue of (16) for subsequent steps. The value $\varepsilon/5$ is heuristic; but, it expresses the basic idea that very good accuracy is required to continue solving an equation by the diagonal approximation.

Normally, as h increases, the matrix $A_n$ becomes less diagonally dominant. Thus, as the integration proceeds, fewer of the equations are solved using (20). However, it is possible for an equation to become diagonally dominant during the integration. There is no easy way to detect this event when it is solved as part of system (16). But a crude check is made by monitoring $E_c$. The condition $E_c < \varepsilon/1000$ shows extreme accuracy, so for the next step all the indices are again placed into L. This allows a new determination of the set of equations to be solved by the diagonal approximation.

The last important part of the integrator is the algorithm for controlling step size. This is based on an estimate of the truncation error and on the convergence error $E_c$, defined by (29).

For simplicity, we use the truncation error form for constant step size, that is,

$$E_T^i = 0.0966\ h^4\ y^{(4)i}(t).$$

The fourth derivative of $Y^i$ is approximated by interpolating a fourth degree polynomial through the values $y_n$, $y_{n-1}$, $y_{n-2}$, $\dot{y}_n$, and $\dot{y}_{n-1}$. The fourth derivative is then given by 4! times the leading coefficient of this polynomial. An rms value for the fourth derivative is given by

$$Y^{(4)}_{rms} = \left( \sum_{i=1}^{N} (y^{(4)i})^2/N \right)^{1/2}.$$

We then define $h_T$ by the relation

$$0.0966\ h_T^4\ \ Y_{rms}^{(4)} = \varepsilon \quad . \qquad (30)$$

The quantity $h_T$ is an upper limit for the next step size.

The step size also depends on the convergence error $E_c$. We define

$$h_c = h_n \left[1.0 + 0.1 \log_{10} (\varepsilon/E_c)\right], \tag{31}$$

and take the minimum of the values $h_T$ and $h_c$ as the next step size $h_{n+1}$.

Formula (31) provides for a slow, steady increase in step size. This approach helps avoid oscillations in the step size. The value of h can increase rapidly only if the predictor and the corrector are in very close agreement.

In summary, the major innovation of the K-integrator is the method of solving the system of linear equations associated with the Jacobian. By using a diagonal approximation, the effort required for this operation can be substantially reduced. To increase the efficiency of this procedure, more effort than usual is invested in obtaining an accurate predicted value. A nonorthodox rational predictor is used, with a further heuristic modification based on the results obtained in the previous steps. The step-size changing algorithm is based on both truncation and convergence errors. It attempts to increase the step size by a small amount each step rather than by a large amount every several steps.

## III. COMPARISONS

The K-integrator has been compared with EPISODE for selected problems. EPISODE can be run with several variants; we use the backward differentiation formulas (BDF), suitable for stiff problems, with a user supplied Jacobian.

Some general observations can be made concerning the two integrators. First, neither one will be efficient if a problem has eigenvalues near the imaginary axis. This area is outside the region of stability for the K-corrector (see page 10) and the BDF formulas. In a problem with eigenvalues $-10 \pm 100\,i$[9] neither method performed adequately. Such examples are not considered further.

Also, since both programs are variable - step size, they should be able to handle discontinuities, by reducing the step size sufficiently. This was one of the reasons for developing EPISODE. This aspect is tested by the last three problems.

---

[9] *W. H. Enright, T. C. Hull, and B. Lindberg,* BIT *15 (1975), 10.*

There are also important differences. The K-method was developed primarily for equations involving chemical reactions. These depend on reaction rates, most of which are known only approximately. Also, measurements of concentrations of species cannot be done to a very high accuracy. So we normally are satisfied with one or two place accuracy, since any further accuracy in solving the equations will not be physically meaningful.

Since the K-integrator is a fixed order method, we expect EPISODE to be more efficient at stricter error tolerances, where truncation error becomes more important. The treatment of the Jacobian is also important. At a stricter error tolerance, more steps will be taken, and the Jacobian will not change as much per step. So EPISODE will in general update the Jacobian less frequently and will become more efficient.

For a given error tolerance, the relative efficiency of the two methods is quite problem dependent. EPISODE is more efficient if the Jacobian changes slowly. Otherwise, the form of the Jacobian is not important. The K-integrator will do better if the Jacobian is diagonally dominant. Since it updates the relevant part of the matrix each step, it is not bothered by a rapidly changing Jacobian. In particular, the K-integrator will tend to be more efficient at the start of an integration, when h is small and at least some of the Y values change rapidly. Since many stiff problems are integrated until a steady state is attained, EPISODE will be more efficient at the end of the integration, when the Y values change slowly.

If the strategies of both the K - and Gear integrators could be successfully combined, the result might lead to a more efficient, general, stiff O.D.E. algorithm.

## IV. NUMERICAL RESULTS

Both the K-integrator and EPISODE have been run for a set of twelve problems, given in Appendix A. The results of the comparisons are in Tables I through IV. All the runs were made on a CDC 7600 in single precision.

We are interested in the efficiency and accuracy of the methods. Efficiency can be measured by the run time. The times given in the tables are the actual times used by the core integrator. The tables also give the number of derivative calls, the number of matrix factorizations, the average dimension of the matrices involved, and the total number of integration steps.

For determining the error, the "correct" answer Y is found using an error criterion of $10^{-8}$ or $10^{-10}$, depending on the problem. The error

Table I. Error Tolerance is $10^{-2}$

| | | RUN TIME | FINAL ERROR | MAX ERROR | DER CALLS | MATRIX FAC | AVERAGE DIM | NO. OF STEPS |
|---|---|---|---|---|---|---|---|---|
| A | E | .039 | .10 | .18 | 75 | 32 | 10.0 | 57 |
| | K | .017 | .00 | .03 | 88 | 0 | 0.0 | 43 |
| B | E | .022 | .19 | 1.21 | 83 | 20 | 6.0 | 54 |
| | K | .022 | .00 | .98 | 130 | 39 | 2.0 | 56 |
| C | E | .041 | .07 | 12.19 | 220 | 28 | 4.0 | 117 |
| | K | .048 | .09 | 17.93 | 394 | 152 | 1.0 | 178 |
| D | E | .003 | .46 | .46 | 18 | 8 | 3.0 | 11 |
| | K | .008 | .00 | .00 | 86 | 10 | 1.0 | 42 |
| E | E | .002 | .01 | .01 | 7 | 6 | 4.0 | 6 |
| | K | * | * | * | * | * | * | * |
| F | E | .088 | .16 | .16 | 251 | 71 | 7.0 | 161 |
| | K | .049 | .58 | .58 | 263 | 62 | 2.3 | 130 |
| G | E | .086 | .00 | .21 | 128 | 54 | 9.0 | 84 |
| | K | .079 | .00 | .01 | 203 | 62 | 5.6 | 99 |
| H | E | 14.604 | .12 | 5.41 | 4323 | 1318 | 26.0 | 1938 |
| | K | 2.062 | .00 | 3.13 | 684 | 254 | 16.9 | 318 |
| I1 | E | 14.870 | .52 | 3.20 | 883 | 144 | 64.0 | 522 |
| | K | 10.662 | .14 | 3.18 | 832 | 278 | 30.7 | 412 |

**See text.*

TABLE II. ERROR TOLERANCE IS $10^{-4}$

| | | RUN TIME | FINAL ERROR | MAX ERROR | DER CALLS | MATRIX FAC | AVERAGE DIM | NO. OF STEPS |
|---|---|---|---|---|---|---|---|---|
| A | E | .082 | .41 | .41 | 179 | 39 | 10.0 | 119 |
| | K | .038 | .07 | .40 | 196 | 0 | 0.0 | 97 |
| B | E | .049 | .35 | 1.90 | 184 | 24 | 6.0 | 115 |
| | K | .039 | .08 | 2.35 | 236 | 85 | 1.6 | 107 |
| C | E | .091 | .56 | 13.05 | 452 | 70 | 4.0 | 248 |
| | K | .123 | .17 | 65.10 | 1012 | 413 | 1.0 | 451 |
| D | E | .008 | .18 | .64 | 48 | 13 | 3.0 | 28 |
| | K | .011 | .05 | .05 | 108 | 19 | 1.6 | 53 |
| E | E | .002 | .67 | .67 | 8 | 6 | 4.0 | 6 |
| | K | .016 | .00 | .00 | 116 | 34 | 2.7 | 57 |
| F | E | .134 | .85 | .85 | 369 | 94 | 7.0 | 225 |
| | K | .094 | 3.34 | 3.34 | 499 | 145 | 1.4 | 247 |
| G | E | .152 | .00 | .27 | 267 | 65 | 9.0 | 172 |
| | K | .119 | .00 | .14 | 288 | 97 | 5.8 | 142 |
| H | E | 75.586 | .00 | 8.76 | 15096 | 7995 | 26.0 | 8934 |
| | K | 3.913 | .05 | 10.00 | 1510 | 504 | 14.8 | 726 |
| I1 | E | 17.015 | 2.98 | 11.22 | 1337 | 137 | 64.0 | 857 |
| | K | 21.911 | .80 | 13.72 | 2022 | 638 | 27.8 | 1004 |

TABLE III. ERROR TOLERANCE IS $10^{-6}$

| | | RUN TIME | FINAL ERROR | MAX ERROR | DER CALLS | MATRIX FAC | AVERAGE DIM | NO. OF STEPS |
|---|---|---|---|---|---|---|---|---|
| A | E | .156 | .23 | .74 | 373 | 42 | 10.0 | 231 |
| | K | .106 | 2.12 | 2.12 | 550 | 0 | 0.0 | 274 |
| B | E | .111 | .07 | 20.15 | 416 | 35 | 6.0 | 268 |
| | K | .095 | 1.20 | 9.44 | 606 | 152 | 1.6 | 287 |
| C | E | .168 | .15 | 5.49 | 798 | 87 | 4.0 | 494 |
| | K | .337 | .68 | 166.79 | 2810 | 1041 | 1.0 | 1318 |
| D | E | .014 | .59 | 1.93 | 88 | 15 | 3.0 | 52 |
| | K | .016 | .47 | .47 | 166 | 39 | 1.9 | 80 |
| E | E | .005 | 1.82 | 1.82 | 23 | 9 | 4.0 | 14 |
| | K | .019 | .19 | .23 | 116 | 34 | 2.7 | 69 |
| F | E | .224 | 1.06 | 1.06 | 613 | 120 | 7.0 | 410 |
| | K | .225 | 12.42 | 12.42 | 1314 | 183 | 1.5 | 654 |
| G | E | .260 | .04 | 1.11 | 502 | 79 | 9.0 | 310 |
| | K | .288 | .00 | .48 | 728 | 224 | 5.9 | 359 |
| H | E | 46.046 | .00 | 19.49 | 10795 | 4461 | 26.0 | 6533 |
| | K | 7.333 | .51 | 29.23 | 3861 | 952 | 12.8 | 1886 |
| I1 | E | 25.651 | 1.19 | 7.00 | 2306 | 177 | 64.0 | 1688 |
| | K | 394.610 | .71 | 44.62 | 15719 | 6578 | 43.5 | 7813 |

TABLE IV. ERROR TOLERANCE IS $10^{-2}$

| | | RUN TIME | FINAL ERROR | MAX ERROR | DER CALLS | MATRIX FAC | AVERAGE DIM | NO. OF STEPS |
|---|---|---|---|---|---|---|---|---|
| I2 | E | 19.650 | .23 | 3.49 | 889 | 208 | 64.0 | 525 |
| | K | 20.159 | .01 | .75 | 976 | 367 | 37.7 | 457 |
| I3 | E | 61.553 | .43 | .43 | 2214 | 713 | 64.0 | 1376 |
| | K | 28.872 | .02 | .23 | 2322 | 907 | 27.1 | 901 |
| I4 | E | 38.228 | .16 | 8.68 | 1313 | 455 | 64.0 | 808 |
| | K | 5.695 | .01 | 2.87 | 779 | 250 | 22.3 | 361 |

may be absolute $\left(\sum_{i=1}^{N} (Y^i-y^i)^2/N\right)^{1/2}$ or relative $\left(\sum_{i=1}^{N} ((Y^i-y^i)/Y^i)/N\right)^{1/2}$.
Error is measured at four times, equally spaced on a logarithmic scale between the initial step size $h_0$ and the final time $t_f$.

Both methods actually control error on a per step basis. The error that we have computed is the global error, which is of more interest to the user. The relation between the user supplied local error tolerance $\varepsilon$ and the global error is highly problem dependent. However, a reliable code should keep global errors below a bound proportional to the local error tolerance, for a given problem. The tables give the error at the end time and the maximum error at the four output times. For convenience, the errors are given in units of the error tolerance.

Appendix B gives a listing of the code for the K-integrator for a sample problem. The output is given at different times and for different error criterions, as described above.

The first five problems are from an article by Enright, Hull, and Lindberg[9]. These are part of a larger set of problems, proposed as a test set for comparing stiff integrators. All the systems are small, but they show that both methods can handle a variety of types of problems. All the problems are run with an absolute error criterion.

The K-method is very efficient for problems A and B, because it can take advantage of the diagonal dominance of the Jacobians. In fact, in solving problem A no matrix factorizations are performed.

For problem C, the K-integrator becomes less efficient and less accurate at $\varepsilon = 10^{-6}$. The difficulty is due to the fact that one component becomes very large, on the order of $10^4$. Since an absolute error criterion is being used, extreme accuracy is required. For this type of problem, the lack of higher order formulas in the K-method can cause difficulties.

For problems D and E, EPISODE is more efficient. The difference here is due to EPISODE's step changing strategy. Both problems are relatively easy, so EPISODE increases the step size rapidly. The K-method has a more conservative algorithm for changing step size, and takes many more steps. However, both programs are extremely fast here.

The K-integrator does fail on problem E for $\varepsilon = 10^{-2}$. Here the problem is caused by the fact that the largest component is of the order $10^{-3}$. So a step can be accepted by the integrator even with no significant digits in the final y values. As a result, the program uses the diagonal approximation too long. The resulting inaccuracies lead to instabilities that do not damp out.

In general, both methods are efficient and reasonably accurate on these problems. However, the K-integrator does experience some difficulties, due to very large or very small components combined with an absolute error criterion, whereas EPISODE does not.

The remaining systems are chemical reaction problems, and here a relative error criterion is used. To prevent control of the step size by species whose concentrations have become negligible, an artificial formation term of $10^{-30}$ is added to $\dot{y}$ every time the derivative subroutine is called. This is also an easy way of preventing underflow.

Problem F is a demonstration reaction set proposed by Edelson[10]. It is a simplified version of an atmospheric chemistry problem. The K-method is faster than EPISODE for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-4}$. The problem is sufficiently diagonally dominant that the K-integrator can just use the diagonal approximation for a good part of the integration, and works with fairly small matrixes even near the end of the integration time. However, at $\varepsilon = 10^{-6}$ its run time is equal to EPISODE, and it is less accurate. Again, the K-method will sometimes experience difficulties at $\varepsilon = 10^{-6}$, due to the lack of higher order formulas.

Problem G and H are simulations of the chemistry in a gun barrel, i.e., under conditions of high temperature and high pressure. The problem is quite stiff.

Problem G involves nine species. Both programs are roughly comparable.

Problem H involves twenty-six species, some of whose concentrations are relatively small. EPISODE experiences major difficulties, due to its step size changing strategy. EPISODE attempts to make large changes in the step size, on the order of 33%. This leads to instability in the minor species. The program cannot meet its error criterion and must reduce the step size. This prevents the step size from increasing normally, and the integration takes a very large number of steps. The K-integrator, with changes of 5% to 10%, can increase its step size consistently.

Problem I1 is an atmospheric model of charge flow under the influence of a large electron flux. The electron density starts at a high level and decays to zero. The systems consist of 64 species. The reactions involved are given by Heimerl and Niles[11].

At $\varepsilon = 10^{-2}$, the K-integrator is somewhat faster. However, the integrators proceeded in very different ways. For example, to reach the third output time, 3.16 seconds, takes EPISODE 12.4 seconds and the K-integrator only 4.5 seconds.

---

[10] *D. Edelson, J. Chem. Ed. 52 (1975), 642.*

[11] *J. M. Heimerl and F. E. Niles, "BENCHMARK -76: Model Computations for Disturbed Atmospheric Conditions. 1. Input Parameters", BRL Report No. 2022, October 1977 (ADA #A050355).*

The K-method is extremely efficient at the start of the integration, where it works with very small matrixes. But by the end of the integration, the K-integrator is working with fifty-four by fifty-four matrixes. Meanwhile, EPISODE updates the Jacobian about every 3.6 steps, and is reasonably efficient throughout the integration. So the efficiency of the programs is highly problem dependent.

At $\varepsilon = 10^{-6}$, the K-integrator is very slow. Because of truncation error, it cannot increase the step size adequately.

The remaining three problems involve discontinuities in the driving function, that is, the electron density, for the above problem. The discontinuities occur at the powers of 10, starting at $10^{-6}$ seconds. For comparison purposes, the equations are integrated out to a time between discontinuities, since the values are somewhat ambiguous at the discontinuities. Results are given only for $\varepsilon = 10^{-2}$.

I2 is a sawtooth driving function. The K-integrator is slightly less efficient here.

I3 is a square wave driving function. The comparison is made at $5 \times 10^{2}$ because EPISODE cannot integrate past the discontinuity at $10^{3}$. Even using the smallest increment possible on a CDC 7600 in single precision, the program cannot meet its error criterion.

The program enters an infinite loop. Using a step size of $4 \times 10^{-12}$, it attempts to cross the discontinuity. Failing its convergence tests it reduces the step size to $2 \times 10^{-12}$. But $10^{3} + 2 \times 10^{-12} = 10^{3}$, and no progress is made. It increases the step size to $3 \times 10^{-12}$, which is still too small an increment to register. Finally, it increases to the value $4 \times 10^{-12}$ again. This process repeats indefinitely.

The reason for this is the way EPISODE determines the step size. EPISODE compares the original predictor with the final accepted y value. At a discontinuity, the original explicit predictor behaves very badly, since it has no information concerning the changed conditions. The step size must be reduced severely to obtain agreement.

The K-integrator, however, compares the modified predictor $y^{P2}$ with the corresponding corrector $y^{C2}$. The existence of the discontinuity is fed in through the Jacobian, and the step size does not have to be reduced so far.

The K-method is also much faster than EPISODE. At the discontinuities, where h is reduced, it can use the simpler diagonal approximation to solve the equations.

Problem I4 is a combination of I2 and I3. The electron density forms a series of ramps, alternately increasing and decreasing, but starting at the same value at each decade. So there is a discontinuous derivative as well as function.

Here EPISODE cannot get past the discontinuity at 1.0 second; so the comparison was made at 0.5 second. Not only is the K-method able to integrate the problem more rapidly than EPISODE, but it also is able to integrate over the entire range of interest, i.e., $10^4$ seconds.

In conclusion the K-integrator does appear to be comparable to the variable-order EPISODE program, at least for the looser error tolerance. It can be more efficient, depending on the problem, and permits integration over some severe functional discontinuities.

## REFERENCES


1. M. D. Kregel and E. L. Lortie, "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations", BRL Report No. 1733, July 1974 (ADA #A003855).

2. M. D. Kregel and J. M. Heimerl, "Comments on the Solutions of Coupled Stiff Differential Equations", BRL Memorandum Report No. 2769, July 1977; or Proceedings of the 1977 Army Numerical Analysis and Computers Conference, ARO Report 77-3, November 1977, pp. 553-563 (ADA #A043122).

3. T. P. Coffee, J. M. Heimerl, and M. D. Kregel, "A Numerical Method for Large Stiff Systems of Ordinary Equations", Transactions of the 24th Conference of Army Mathematicians, ARO Report 79-1, January 1977, pp. 249-257.

4. G. G. Dahlquist, AMS Symp. Appl. Math. 15, (1963), 147.

5. C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, 1971.

6. G. D. Byrne and A. C. Hindmarsh, ACM Trans. Math. Software (1975), 71.

7. G. K. Gupta, Math. of Comp. 30 (1976), 417.

8. O. B. Widlund, BIT 7 (1967), 65.

9. W. H. Enright, T. C. Hull, and B. Lindberg, BIT 15 (1975), 10.

10. D. Edelson, J. Chem. Ed. 52 (1975), 642.

11. J. M. Heimerl and F. E. Niles, "BENCHMARK -76: Model Computations for Disturbed Atmospheric Conditions. 1. Input Parameters", BRL Report No. 2022, October 1977 (ADA # A050355).

## APPENDIX A. SPECIFICATION OF PROBLEMS

Listed below are the systems of differential equations used in the tests. In all cases the initial time $t_0 = 0$. The final time $t_f$ and the initial step size $h_0$ are given. The last four systems are not specified in detail because of their complexity.

A. Linear with real eigenvalues

$$\dot{Y}^i = i^5 Y^i \quad Y^i(0) = 1 \quad i = 1,2,\ldots,10$$

$$t_f = 1 \qquad h_0 = 10^{-5}$$

B. Linear with non-real eigenvalues.

$$\dot{Y}^1 = -10Y^1 + 25\ Y^2 \qquad Y^1(0) = 1$$

$$\dot{Y}^2 = -25Y^1 - 10\ Y^2 \qquad Y^2(0) = 1$$

$$\dot{Y}^3 = -\ 4Y^3 \qquad Y^3(0) = 1$$

$$\dot{Y}^4 = -\ Y^4 \qquad Y^4(0) = 1$$

$$\dot{Y}^5 = -\ 0.5\ Y^5 \qquad Y^5(0) - 1$$

$$\dot{Y}^6 = -\ 0.1\ Y^6 \qquad Y^6(0) = 1$$

$$t_f = 20 \qquad h_0 = 10^{-2}$$

C. Non-linear coupling.

$$\dot{Y}^1 = -\ Y^1 + 2 \qquad Y^1(0) = 1$$

$$\dot{Y}^2 = -\ 10\ Y^2 + 20\ (Y^1)^2 \qquad Y^2(0) = 1$$

$$\dot{Y}^3 = -\ 40\ Y^3 + 80\ [(Y^1)^2 + (Y^2)^2] \qquad Y^3(0) = 1$$

$$\dot{Y}^4 = -100\ Y^4 + 200[(Y^1)^2 + (Y^2)^2 + (Y^3)^2] \quad Y^4(0) = 1$$

$$t_f = 20 \qquad h_0 = 10^{-2}$$

D. Non-linear with real eigenvalues.

$$\dot{Y}^1 = -Y^1 + 10^8 Y^3 (1 - Y^1) \qquad Y^1(0) = 1$$

$$\dot{Y}^2 = -10 Y^2 + 3 \times 10^7 Y^3 (1 - Y^2) \qquad Y^2(0) = 0$$

$$\dot{Y}^3 = -\dot{Y}^1 - \dot{Y}^2 \qquad Y^3(0) = 0$$

$$t_f = 1 \qquad h_0 = 3.3 \times 10^{-8}$$

E. Non-linear with non-real eigenvalues.

$$\dot{Y}^1 = -7.89\times10^{-10} Y^1 - 1.1\times10^7 Y^1 Y^3 \qquad Y^1(0) = 1.76\times10^{-3}$$

$$\dot{Y}^2 = 7.89\times10^{-10} Y^1 - 1.13\times10^9 Y^2 Y^3 \qquad Y^2(0) = 6$$

$$\dot{Y}^3 = 7.89\times10^{-10} Y^1 - 1.1\times10^7 Y^1 Y^3 \qquad Y^3(0) = 0$$
$$+ 1.13\times10^3 Y^4 - 1.13\times10^9 Y^2 Y^3$$

$$\dot{Y}^4 = 1.1\times10^7 Y^1 Y^3 - 1.13\times10^3 Y^4 \qquad Y^4(0) = 0$$

$$t_f = 1000 \qquad h_0 = 5 \times 10^{-5}$$

F. Atmospheric reaction set. A set of 7 chemical species and 7 reactions.

$$t_f = 1000 \qquad h_0 = 10^{-16}$$

G. Hot Gas. A set of 9 chemical species and 57 reactions under high temperature and pressure.

$$t_f = 10^{-5} \qquad h_0 = 10^{-15}$$

H. Expanded Hot Gas. A set of 26 chemical species and 227 reactions under high temperature and pressure.

$$t_f = 10^{-5} \qquad h_0 = 10^{-15}$$

I. BENCHMARK - 76. A set of 64 chemical species and 498 reactions simulating chemistry in the upper atmosphere. The reactions are driven by the electron density.

1. Relaxation of driving force.

   $t_f = 10^4$ $\qquad$ $h_0 = 10^{-10}$

2. Saw tooth drawing force (discontinuous derivative).

   $t_f = 5 \times 10^3$ $\qquad$ $h_0 = 10^{-10}$

3. Square Wave driving force (discontinuous function).

   $t_f = 5 \times 10^2$ $\qquad$ $h_0 = 10^{-10}$

4. Ramp driving force (discontinuous function and derivative).

   $t_f = 0.5$ $\qquad$ $h_0 = 10^{-10}$

## APPENDIX B. PROGRAM LISTING

A listing of the computer code follows. The code is set-up to solve problem C, page 27.

```
      PROGRAM K1NT(OUTPUT,TAPE6=OUTPUT)                                 MAIN       2
      D1MENSION YC(30)                                                  MAIN       3
C     THE MAIN PROGRAM AND THE SUBROUTINE DER ARE SUPPLIED              MAIN       4
C     BY THE USER.                                                      MAIN       5
C     AS AN EXAMPLE                                                     MAIN       6
C     SOLVE SYSTEM C5    ENRIGHT, HULL, AND LINDBERG.                   MAIN       7
C     DY1/DT=-Y1+2                                                      MAIN       8
C     DY2/DT=-10*Y2+20*Y1**2                                            MAIN       9
C     DY3/DT=-40*Y3+80*(Y1**2+Y2**2)                                    MAIN      10
C     DY4/DT=-100*Y4+200*(Y1**2+Y2**2+Y3**2)                            MAIN      11
C     Y1(0)=1.0    Y2(0)=1.0    Y3(0)=1.0    Y4(0)=1.0                  MAIN      12
      D1MENS1ON Y(30)                                                   MAIN      13
      COMMON/TAB1/T1N,H1N,TEND,T,EC                                     MAIN      14
      COMMON/TAB3/ND1M,1D1M                                             MAIN      15
      COMMON/TAB4/N1TG,NDER,NJAC,NM1,NDM1                               MAIN      16
C     SET 1NPUT DATA.                                                   MAIN      17
C     1D1M=D1MENS1ON OF VECTORS AND MATR1CES.                           MAIN      18
C     ND1M=NUMBER OF UNKNOWNS.                                          MAIN      19
C     T1N=START T1ME                                                    MAIN      20
C     H1N=1N1T1AL STEP SIZE                                             MAIN      21
C     TEND=END T1ME                                                     MAIN      22
C     T=CURRENT T1ME                                                    MAIN      23
C     EC=ERROR CR1TER1ON                                                MAIN      24
      DO 1000 KTOUT=1,4                                                 MAIN      25
      H1N=1.0E-2                                                        MAIN      26
      TEND=20.                                                          MAIN      27
      RED=FLOAT(KTOUT)*ALOG10(TEND)/4.0+                                MAIN      28
     * (4.0-FLOAT(KTOUT))*ALOG10(HIN)/4.0                              MAIN      29
      TEND=10.**RED                                                     MAIN      30
      DO 1000 KEC=4,10,2                                                MAIN      31
      KB=12-KEC                                                         MAIN      32
      1D1M=30                                                           MAIN      33
      ND1M=4                                                            MAIN      34
      TIN=0.0                                                           MAIN      35
      EC=.1**KB                                                         MAIN      36
      WR1TE(6,1010)EC                                                   MAIN      37
 1010 FORMAT(///2X,10HERR CR1T =,1PE12.4)                               MAIN      38
      CHECK=TEND-TEND*1.0E-10                                           MAIN      39
C     1ERROR=1 MEANS ABSOLUTE ERROR CR1TER1ON.                          MAIN      40
C     IERROR=2 MEANS RELATIVE ERROR CRITERION.                          MAIN      41
      1ERROR=1                                                          MAIN      42
C     SET THE INITIAL VALUES.                                           MAIN      43
      Y(1)=1.0                                                          MAIN      44
      Y(2)=1.0                                                          MAIN      45
      Y(3)=1.0                                                          MAIN      46
      Y(4)=1.0                                                          MAIN      47
C     SET THE COUNTERS TO 0                                             MAIN      48
C     NITG=NO. OF INTEGRATION STEPS.                                    MAIN      49
C     NDER=NO. OF DERIVATIVE EVALUATIONS                                MAIN      50
C     NJAC=NO. OF JACOBIAN EVALUATIONS                                  MAIN      51
C     NM1=NO. OF MATRIX INVERSIONS                                      MAIN      52
C     NDM1=SUM OF THE DIMENSIONS OF THE MATRICES INVERTED               MAIN      53
      N1TG=0                                                            MAIN      54
      NDER=0                                                            MAIN      55
      NJAC=0                                                            MAIN      56
      NMI=0                                                             MAIN      57
      NDMI=0                                                            MAIN      58
```

```
                FT=SECONO(CP)                                                   MAIN      59
                00 4 M=1,100000                                                 MAIN      60
60          C   CALL THE INTEGRATION SUBROUTINE.                                MAIN      61
                CALL MRK(Y,SING,IERROR)                                         MAIN      62
                IF(T.GT.CHECK)GOTO 30                                           MAIN      63
                IF(SING.GT.0.5)GOTO 30                                          MAIN      64
              4 CONTINUE                                                        MAIN      65
65          C   RRINT OESIREO OUTPUT OATA.                                      MAIN      66
             30 GT=SECONO(CP)                                                   MAIN      67
                HT=GT-FT                                                        MAIN      68
                WRITE(6,500)HT                                                  MAIN      69
            500 FORMAT(1H0,10HRUN TIME =,1PE12.4)                               MAIN      70
70              IF(NMI.EQ.0)GOTO 35                                             MAIN      71
                AO=FLOAT(NOMI)/FLOAT(NMI)                                       MAIN      72
             35 WRITE(6,510)NITG,NOER,NJAC                                      MAIN      73
            510 FORMAT(1H0,11HMRK CALLS =,I6,6X,18HDERIVATIVE CALLS =,          MAIN      74
               *I6,6X,16HJACOBEAN CALLS =,I6)                                   MAIN      75
75              WRITE(6,520)NMI,AO                                              MAIN      76
            520 FORMAT(1H0,26HNO. OF MATRIX INVERSIONS =,I6,6X,                 MAIN      77
               * 19HAVERAGE DIMENSION =,1PE12.4)                                MAIN      78
                WRITE(6,505)(Y(K),K=1,NOIM)                                     MAIN      79
            505 FORMAT(1H0,1R5E20.12)                                           MAIN      80
80              WRITE(6,550)T                                                   MAIN      81
            550 FORMAT(1H0,12HFINAL TIME =,1PE12.4)                             MAIN      82
                IF(KEC.GT.4)GO TO 800                                           MAIN      83
                00 810 K=1,NDIM                                                 MAIN      84
            810 YC(K)=Y(K)                                                      MAIN      85
85          800 ERR=0.0                                                         MAIN      86
                DO 850 K=1,NOIM                                                 MAIN      87
                IF(IERROR.EQ.1)OENOM=1.0                                        MAIN      88
                IF(IERROR.EQ.2)OENOM=YC(K)                                      MAIN      89
                IF(OENOM.EQ.0.0)GO TO 1000                                      MAIN      90
90          850 ERR=ERR+((YC(K)-Y(K))/OENOM)**2                                 MAIN      91
                ERR=SQRT(ERR/FLOAT(NOIM))                                       MAIN      92
                WRITE(6,880)ERR                                                 MAIN      93
            880 FORMAT(/5X,7HERROR =,1PE12.4)                                   MAIN      94
           1000 CONTINUE                                                        MAIN      95
95           99 STOR                                                            MAIN      96
                ENO                                                             MAIN      97
```

```
      SUBROUTINE MRK(Y,SING,IERROR)                                    MRK     2
C     THIS VERSION USED IN ARTICLE AND BRL REPORT                      MRK     3
C     A NUMERICAL METHOD TO INTEGRATE LARGE STIFF SYSTEMS OF ODES.     MRK     4
C  CREATED SEPTEMBER 22, 1978.                                         MRK     5
      DIMENSION Y(30),Z(30),W(30),G(30)                                MRK     6
      DIMENSION F0(30),F1(30),F2(30),R0(30),R1(30),R2(30)              MRK     7
      DIMENSION ALF(30),DERY(30)                                       MRK     8
      DIMENSION AUX2(30),AUX3(30),AUX4(30),AUX5(30),AUX6(30)           MRK     9
      DIMENSION AUX7(30),AUX8(30),AUX10(30),AUX11(30),AUX12(30)        MRK    10
      DIMENSION AE(1)                                                  MRK    11
      COMMON/TAB1/TIN,HIN,TEND,T,EC                                    MRK    12
      COMMON/TAB2/A(30,30),KEY(30)                                     MRK    13
      COMMON/TAB3/NDIM,IDIM                                            MRK    14
      COMMON/TAB4/NITG,NOER,NJAC,NMI,NOMI                              MRK    15
      EQUIVALENCE(AE(1),A(1,1))                                        MRK    16
      LOGICAL LOGIC1,LOGIC2,LOGIC3                                     MRK    17
C     THE PROGRAM CAN BE RUN IN EITHER SINGLE OR DOUBLE PRECISION      MRK    18
C     BY CHANGING THE DEFINITION OF THE FOLLOWING FUNCTIONS.           MRK    19
      ZABS(S)=ABS(S)                                                   MRK    20
C     ZABS(S)=DABS(S)                                                  MRK    21
      ZFLOAT(L)=FLOAT(L)                                               MRK    22
C     ZFLOAT(L)=DBLE(FLOAT(L))                                         MRK    23
      ZLOG10(S)=ALOG10(S)                                              MRK    24
C     ZLOG10(S)=DLOG10(S)                                              MRK    25
      Z2MAX(S,T)=AMAX1(S,T)                                            MRK    26
C     Z2MAX(S,T)=DMAX1(S,T)                                            MRK    27
      Z2MIN(S,T)=AMIN1(S,T)                                            MRK    28
C     Z2MIN(S,T)=DMIN1(S,T)                                            MRK    29
      Z3MAX(S,T,U)=AMAX1(S,T,U)                                        MRK    30
C     Z3MAX(S,T,U)=DMAX1(S,T,U)                                        MRK    31
      Z3MIN(S,T,U)=AMIN1(S,T,U)                                        MRK    32
C     Z3MIN(S,T,U)=DMIN1(S,T,U)                                        MRK    33
      ZSQRT(S)=SQRT(S)                                                 MRK    34
C     ZSQRT(S)=DSQRT(S)                                                MRK    35
      IF(NITG.GT.0)GOTO 7                                              MRK    36
C  *  *  *  *  *  *                                                    MRK    37
C  *  *  *  *  *  *                                                    MRK    38
C  SET INITIAL VALUES.                                                 MRK    39
      LOGIC1=.FALSE.                                                   MRK    40
      LOGIC2=.TRUE.                                                    MRK    41
      LOGIC3=.FALSE.                                                   MRK    42
      ACC1=EC*1.0E-2                                                   MRK    43
      ACC2=EC*1.0E-4                                                   MRK    44
      ACC3=EC*0.1                                                      MRK    45
      NUPD=1                                                           MRK    46
      KEND=0                                                           MRK    47
      H=HIN                                                            MRK    48
      HP=H*1.0E5                                                       MRK    49
      T=TIN                                                            MRK    50
      NP1=IDIM+1                                                       MRK    51
      CALL DER(Y,DERY)                                                 MRK    52
      DO 1 J=1,NDIM                                                    MRK    53
      AUX12(J)=DERY(J)                                                 MRK    54
      AUX11(J)=Y(J)                                                    MRK    55
      KEY(J)=0                                                         MRK    56
    1 ALF(J)=0.0                                                       MRK    57
      HOLD=1.0E34                                                      MRK    58
```

```
                  CALL JACOB                                                        MRK        59
                  DO 55 J=1,ND1M                                                    MRK        60
60                1J=(J-1)*1DIM                                                     MRK        61
                  RO( J ) = ZABS(  AE( 1J+J ) / Y( J ) )                            MRK        62
                  COO=.01/(RO(J)+1.0E-34)                                           MRK        63
                  IF(HOLD.GT.COO) HOLD=COO                                          MRK        64
               55 CONTINUE                                                          MRK        65
65                IF(H.GT.HOLD) H=HOLD                                              MRK        66
                  PRINT 27,HOLD                                                     MRK        67
               27 FORMAT ( 47H  THE UPPER BOUND FOR THE INITIAL STEP SIZE IS ,E12.4) MRK        68
            C  *  *  *  *  *  *                                                     MRK        69
            C  *  *  *  *  *  *                                                     MRK        70
70          C  GENERATION OF THE PREDICTOR AND CORRECTOR VALUES.                    MRK        71
                7 IF(NITG.GT.1)GOTO 6                                               MRK        72
            C  GENERATION OF THE COEFFICIENTS USED IN THE                           MRK        73
            C  PREDICTOR-CORRECTOR FORMULA.                                         MRK        74
                  AL=1.0                                                            MRK        75
75                BETA=0.0                                                          MRK        76
                  IF(NITG.EQ.1)AL=0.6                                               MRK        77
                  IF(NITG.EQ.1)BETA=0.4                                             MRK        78
                  A2=0.0                                                            MRK        79
                  B1=0.0                                                            MRK        80
80                DET=HP*(HP+H+H)                                                   MRK        81
                  B=H*H/DET                                                         MRK        82
                  BU=-B*BETA                                                        MRK        83
                  AU=AL+(1.0+B)*BETA                                                MRK        84
                  CU=AL*H/2.0                                                       MRK        85
85                DU=CU+BETA*H*HP*(H+HP)/DET                                        MRK        86
                  GOTO 9                                                            MRK        87
                6 CU=0.35                                                           MRK        88
                  B1=0.05                                                           MRK        89
                  P=HP/H                                                            MRK        90
90                Q=(HP+HPP)/H                                                      MRK        91
                  TPP=3.0+P+P                                                       MRK        92
                  PQ=P*Q                                                            MRK        93
                  BP=1.0+P-B1*P*P                                                   MRK        94
                  DU=(BP+Q+PQ*(1.0+B1-CU))/(TPP+Q+Q+PQ)                             MRK        95
95                A2=(BP-DU*TPP)/(Q*Q*(P-Q))                                        MRK        96
                  BU=(B1+CU-A2*Q+DU-1.0)/P                                          MRK        97
                  AU=1.0-BU-A2                                                      MRK        98
                  DU=DU*H                                                           MRK        99
                  CU=CU*H                                                           MRK       100
100               B1=B1*H                                                           MRK       101
            C GENERATION OF THE PREDICTOR VALUES IN AUX5.                           MRK       102
            C  THE PREDICTOR IS STORED IN W.                                        MRK       103
            C  ALF IS A CORRECTION FACTOR BASED ON PAST DIFFERENCES                 MRK       104
            C  BETWEEN THE PREDICTOR AND THE FINAL ACCEPTED VALUE.                  MRK       105
105             9 IF(NITG.LT.5)GOTO 36                                              MRK       106
                  CALL PDTR(C00,C01,C02,DU,H,HP,HPP)                                MRK       107
                  DO 38 J=1,ND1M                                                    MRK       108
                  AUX3(J)=AU*AUX11(J)+BU*AUX4(J)+A2*AUX10(J)+CU*AUX12(J)+B1*AUX2(J) MRK       109
                  F=F0(J)+F1(J)*C01+F2(J)*C02                                       MRK       110
110               R=R0(J)+R1(J)*C01+R2(J)*C02                                       MRK       111
                  W(J)=(AUX3(J)*C00+DU*F)/(C00+DU*R)                                MRK       112
               38 AUX5(J)=W(J)*(1.0+ALF(J))                                         MRK       113
                  GOTO 39                                                           MRK       114
               36 DO 41 J=1,NDIM                                                    MRK       115
```

```
      AUX3(J)=AU*AUX11(J)+BU*AUX4(J)+A2*AUX10(J)+CU*AUX12(J)+B1*AUX2(J)   MRK  116
   41 AUX5(J)=AUX11(J)+H*AUX12(J)                                         MRK  117
   39 T=T+H                                                               MRK  118
      CALL OER(AUX5,OERY)                                                 MRK  119
C GENERATION OF THE CORRECTOR VALUE, IN Y.                                MRK  120
      DO 34 J=1,NUIM                                                      MRK  121
   34 Y(J)=AUX3(J)+OU*OERY(J)                                             MRK  122
C  *  *  *  *  *  *                                                      MRK  123
C  *  *  *  *  *  *                                                      MRK  124
C  NEWTON-RAPHSON ITERATION.                                              MRK  125
C  IF KEY(J)=0. THE CORRESPONDING Y VALUE IS ACCEPTED                     MRK  126
C  IN THE INITIAL SCREEN.                                                 MRK  127
      DO 100 III=1,10                                                     MRK  128
      CO1=1.0/DU                                                          MRK  129
      CALL JACOB                                                          MRK  130
      ITEST=0                                                             MRK  131
      K=1                                                                 MRK  132
      DO 13 J=1,NDIM                                                      MRK  133
      AU=AUX5(J)                                                          MRK  134
      AE(K)=AE(K)-AU/DU                                                   MRK  135
      BOLD=AU-Y(J)                                                        MRK  136
      IF(KEY(J).EQ.1) ITEST=ITEST+1                                       MRK  137
      Z(J)=BOLD*CO1                                                       MRK  138
      G(J)=Z(J)/AE(K)                                                     MRK  139
   13 K=K+NPI                                                             MRK  140
      IF(ITEST.EQ.0.OR.NITG.EQ.0)GOTO 33                                  MRK  141
      CALL JAC                                                            MRK  142
C THE FOLLOWING BLOCK UPDATES THE Z ARRAY FOR THOSE Y VALUES WHICH HAVE   MRK  143
C CONVERGED.                                                              MRK  144
      DO 116 J=1,NDIM                                                     MRK  145
      IF(KEY(J).EQ.1)GOTO 116                                             MRK  146
      J1=(J-1)*IDIM                                                       MRK  147
      SUBJ=G(J)                                                           MRK  148
      DO 15 I=1,NDIM                                                      MRK  149
      J1=J1+1                                                             MRK  150
      Z(I)=Z(I)-AE(JI)*SUBJ                                               MRK  151
   15 CONTINUE                                                            MRK  152
  116 CONTINUE                                                            MRK  153
C  ANY Y VALUES THAT HAVE NOT CONVERGED ARE FOUND BY EXPLICITLY           MRK  154
C  SOLVING THE SYSTEM OF EQUATIONS.  THE FINAL PREDICTED AND              MRK  155
C  CORRECTED VALUES ARE GENERATED.                                        MRK  156
    4 IF(ITEST.GT.0) CALL FIX (Z,G,SING)                                  MRK  157
      NMI=NMI+1                                                           MRK  158
      NDMI=NDMI+ITEST                                                     MRK  159
      IF (SING.GT.HALF)RETURN                                             MRK  160
   33 DO 12 J=1,NDIM                                                      MRK  161
   12 AUX5(J)=AUX5(J)*(1.0+G(J))                                          MRK  162
      CALL DER(AUX5,OERY)                                                 MRK  163
C  *  *  *  *  *  *                                                      MRK  164
C  *  *  *  *  *  *                                                      MRK  165
C  FIND A LIMIT ERR ON THE TRUNCATION ERROR.                              MRK  166
C  FIND THE CONVERGENCE ERROR DELT.                                       MRK  167
   37 IF(NITG.GE.2)CALL CONT(H,HP,0.00,Z1,G1,Z2)                          MRK  168
      ERR=1.0E-34                                                         MRK  169
      DELT=0.0                                                            MRK  170
      DO 442 J=1,NDIM                                                     MRK  171
      BOLD=AUX3(J)+DU*DERY(J)                                             MRK  172
```

```
                Y(J)=BOLD                                                       MRK       173
                AU=AUX5(J)                                                      MRK       174
                AUX8(J)=0.0                                                     MRK       175
175             IF(NITG.LT.2)GOTO 137                                           MRK       176
                HOLO=Z1*AUX11(J)+Z2*AUX4(J)+QO*DERY(J)+Q1*AUX12(J)              MRK       177
                IF(IERROR.EQ.1)AUX8(J)=HOLO+O*AU                                MRK       178
              C  AUX8 = FOURTH DERIVATIVE / 24                                  MRK       179
                IF(IERROR.EQ.2)AUX8(J)=HOLO/AU+O                                MRK       180
180           C  AUX8 = FOURTH DERIVATIVE / (24*Y)                              MRK       181
              137 IF(LOGIC2)GOTO 42                                             MRK       182
                CO0=AUX6(J)                                                     MRK       183
                CO1=AUX7(J)                                                     MRK       184
                CO2=AUX8(J)                                                     MRK       185
185             AVE=ZABS(CO0+CO1+CO2)/3.+1.0E-34                                MRK       186
                HOLD=ZABS(CO2)/(1.0+(Z3MAX(CO0,CO1,CO2)                         MRK       187
               * - Z3MIN( CO0, CO1, CO2 ) ) / AVE )                             MRK       188
                ERR=ERR+HOLO**2                                                 MRK       189
               42 IF(IERROR.EQ.1)DENOM=2.                                       MRK       190
190             IF(IERROR.EQ.2)DENOM=AU+BOLD                                    MRK       191
              C  DETERMINE WHAT Y VALUES WILL BE ACCEPTED IN THE NEXT STEP.     MRK       192
                ER =ZABS((AU-BOLO)/DENOM)                                       MRK       193
                TEST=ACC1                                                       MRK       194
                IF(III.GT.1.AND.NUPO.EQ.0)TEST=ACC1*0.01                        MRK       195
195             IF(ER.GT.TEST)KEY(J)=1                                          MRK       196
                IF(KEY(J).EQ.1)GO TO 442                                        MRK       197
                IF(IERROR.EQ.2)GO TO 442                                        MRK       198
              C  EVEN FOR AN ABSOLUTE ERROR CRITERION, A MINIMUM RELATIVE       MRK       199
              C  ACCURACY IS NECESSARY TO PREVENT INSTABILITY.                  MRK       200
200             IF(ER.GT.0.05*ABS(AU))KEY(J)=1                                  MRK       201
              442   DELT=DELT+ER**2                                           MRK       202
                OELT=ZSQRT(DELT/ZFLOAT(NOIM))*2.                                MRK       203
                ERR=ZSQRT(ERR/ZFLOAT(NDIM))                                     MRK       204
                IMINN=1                                                         MRK       205
205             IF(NITG.EQ.0)IMINN=2.                                           MRK       206
                IF( III.LE. IMINN ) SDELT=DELT                                  MRK       207
                HLIM=ZSQRT(ZSQRT(EC/ERR))*0.8                                   MRK       208
                HLIM=Z2MAX(HLIM,0.5*H)                                          MRK       209
                IF(III.LT.IMINN) GOTO 100                                       MRK       210
210             IF(DELT.LT.ACC3.AND.H.LE.1.35*HLIM)GO TO 88                     MRK       211
                IF(III.EQ.10)GOTO 44                                            MRK       212
                IF(DELT.LT.ACC3.AND.H.GT.1.35*HLIM)GO TO 44                     MRK       213
                GOTO 100                                                        MRK       214
               44 HOLO=Z2MIN(HLIM,0.75*H)                                       MRK       215
215             T=T-H                                                           MRK       216
                RHP=(HOLO/H)**3                                                 MRK       217
                H=HOLD                                                          MRK       218
                KEND=0                                                          MRK       219
                DO 3 J=1,NDIM                                                   MRK       220
220              3 ALF(J)=ALF(J)*RHP                                            MRK       221
                GOTO 7                                                          MRK       222
              100 CONTINUE                                                      MRK       223
              C  *  *  *  *  *  *                                               MRK       224
              C  *  *  *  *  *  *                                               MRK       225
225           C  CHOOSE THE NEXT H VALUE.                                       MRK       226
              C  RESET THE VALUES OF THE VARIABLES IN PREPAPATION               MRK       227
              C  FOR THE NEXT MRK CALL.                                         MRK       228
                88 SDELT=SDELT+EC*1.0E-20                                       MRK       229
```

```
                HPP=HP                                                          MRK      230
                HP=H                                                            MRK      231
230             H=H*(I.0+0.I2*ZLOGI0(EC/(I2.5*SOELT)))                          MRK      232
                IF(H.LT.HP)H=Z2MIN(H,0.85*HP)                                   MRK      233
                H=Z2MIN(H,HLIM)                                                 MRK      234
              C  IF WE ARE NEAR THE REQUESTEO OUTPUT TIME.                      MRK      235
235           C  CHOOSE THE STEP SIZES SO THE OUTPUT TIME IS REACHED EXACTLY.   MRK      236
                IF(T+5.0*H.LT.TENO)GOTO 65                                      MRK      237
                IF(H.LT.HP)KENO=0                                               MRK      238
                H=HP                                                            MRK      239
                IF(KENO.EQ.1)GOTO 65                                            MRK      240
240             KENO=I                                                          MRK      241
                H=(TENO-T)/5.0                                                  MRK      242
               65 CONTINUE                                                      MRK      243
                CALL JACOB                                                      MRK      244
                RHP=(H/HP)**3                                                   MRK      245
245             SFAC=0.5*Z2MIN(I.0,RHP)                                         MRK      246
                TFAC=SFAC                                                       MRK      247
                K=I                                                             MRK      248
                00 40 J=I,NDIM                                                  MRK      249
                AU=AUX5(J)                                                      MRK      250
250             AUX6(J)=AUX7(J)                                                 MRK      25I
                AUX7(J)=AUX8(J)                                                 MRK      252
                F2(J)=FI(J)                                                     MRK      253
                FI(J)=FO(J)                                                     MRK      254
                FO(J)=OERY(J)-AE(K)                                             MRK      255
                R2(J)=RI(J)                                                     MRK      256
255             RI(J)=RO(J)                                                     MRK      257
                RO(J)=-AE(K)/AU                                                 MRK      258
                AUX2(J)=AUXI2(J)                                                MRK      259
                AUXI2(J)=OERY(J)                                                MRK      260
260             AUXI0(J)=AUX4(J)                                                MRK      26I
                AUX4(J)=AUX11(J)                                                MRK      262
                AUXII(J)=Y(J)                                                   MRK      263
                IF(LOGIC3)ALF(J)=(AU/W(J)-1.0)*SFAC+ALF(J)*TFAC                 MRK      264
               40 K=K+NP1                                                       MRK      265
265             NITG=NITG+1                                                     MRK      266
                IF(NITG.GE.2)LOGICI=.TRUE.                                      MRK      267
                IF(NITG.GE.4)LOGIC2=.FALSE.                                     MRK      268
                IF(NITG.GE.5)LOGIC3=.TRUE.                                      MPK      269
              C  *  *  *  *  *  *                                               MRK      270
270           C  *  *  *  *  *  *                                               MRK      27I
                NUPD=0                                                          MRK      272
                IF(NITG.LE.5)RETURN                                             MRK      273
                IF(SOELT.GT.ACC2)RETURN                                         MRK      274
                IF(KEND.EQ.I)PETURN                                             MRK      275
275             NUPO=I                                                          MRK      276
                00 45 J=I,NDIM                                                  MRK      277
               45  KEY(J)=0                                                    MRK      278
                RETURN                                                          MRK      279
                ENO                                                             MRK      280
```

```
      SUBROUTINE FIX(Z,G,SING)                                          FIX        2
C  CALLED BY MRK.                                                       FIX        3
C  FIX GENERATES THE REDUCED MATRIX.                                     FIX        4
      DIMENSION AE(1),Z(1),G(1)                                         FIX        5
      DIMENSION N(30),Y(30)                                             FIX        6
      COMMON/TAB2/A(30,30),KEY(30)                                      FIX        7
      COMMON/TAB3/NDIM,IDIM                                             FIX        8
      EQUIVALENCE(AE(1),A(1,1))                                         FIX        9
      K=0                                                               FIX       10
      DO 3 J=1,NDIM                                                     FIX       11
      IF(KEY(J).LT.1)GOTO 3                                             FIX       12
      K=K+1                                                             FIX       13
      N(K)=J                                                            FIX       14
      Y(K)=Z(J)                                                         FIX       15
    3 CONTINUE                                                          FIX       16
      JI=0                                                              FIX       17
      DO 2 J=1,K                                                        FIX       18
      L=N(J)                                                            FIX       19
      LM=(L-1)*IDIM                                                     FIX       20
      LML=LM+L                                                          FIX       21
      DO 1 I=1,K                                                        FIX       22
      JI=JI+1                                                           FIX       23
      LCM=LM+N(I)                                                       FIX       24
      AE(JI)=AE(LCM)                                                    FIX       25
    1 CONTINUE                                                          FIX       26
2     CONTINUE                                                          FIX       27
      J=N(1)                                                            FIX       28
      IF(K.EQ.1)G(J)=Y(1)/AE(1)                                         FIX       29
      IF(K.GT.1) CALL MATSDL(K,SING,Y,G,N)                              FIX       30
      RETURN                                                            FIX       31
      END                                                               FIX       32
```

```
      SUBROUTINE MATSOL(N,ERROR,Y,V,NA)                                 MATSOL     2
C     CALLED BY FIX.                                                    MATSOL     3
C  MATSOL PERFORMS A LOWER-UPPER DECOMPOSITION AND SOLVES               MATSOL     4
C  THE SYSTEM OF EQUATIONS BY BACK SUBSTITUTION.                        MATSOL     5
C  PIVOTING IS DONE BY COLUMNS BECAUSE THE MODIFIED FORM                MATSOL     6
C  OF THE JACOBIAN USED RESULTS IN WIDELY DIFFERENT ORDERS              MATSOL     7
C  OF MAGNITUDE IN THE COLUMNS.                                         MATSOL     8
      DIMENSION Y(1),V(1),NA(1),IX(30)                                  MATSOL     9
      COMMON/TAB2/A(900),KEY(30)                                        MATSOL    10
      INTEGER P,Q,PIVOT                                                 MATSOL    11
      ZABS(X)=ABS(X)                                                    MATSOL    12
C     ZABS(X)=DABS(X)                                                   MATSOL    13
C  PERFORM THE LOWER-UPPER DECOMPOSITION.                               MATSOL    14
      ERROR=0.0                                                         MATSOL    15
      DO 1 I=1,N                                                        MATSOL    16
    1 IX(I)=I                                                           MATSOL    17
      DO 8 PIVOT=1,N                                                    MATSOL    18
      BIG=0.0                                                           MATSOL    19
      K=PIVOT+PIVOT*N-N                                                 MATSOL    20
      DO 2 J=PIVOT,N                                                    MATSOL    21
      T=ZABS(A(K))                                                      MATSOL    22
      K=K+N                                                             MATSOL    23
      IF (BIG.GT.T) GOTO 2                                              MATSOL    24
      BIG=T                                                             MATSOL    25
      Q=J                                                               MATSOL    26
    2 CONTINUE                                                          MATSOL    27
      IF (BIG) 3,10,3                                                   MATSOL    28
    3 IF (Q.EQ.PIVOT) GOTO 5                                            MATSOL    29
      II=N*PIVOT-N                                                      MATSOL    30
      JJ=N*Q-N                                                          MATSOL    31
      DO 4 I=1,N                                                        MATSOL    32
      II=II+I                                                           MATSOL    33
      JJ=JJ+I                                                           MATSOL    34
      T=A(II)                                                           MATSOL    35
      A(II)=A(JJ)                                                       MATSOL    36
    4 A(JJ)=T                                                           MATSOL    37
      IT=IX(PIVOT)                                                      MATSOL    38
      IX(PIVOT)=IX(Q)                                                   MATSOL    39
      IX(Q)=IT                                                          MATSOL    40
    5 IF (PIVOT.EQ.N) GOTO 9                                            MATSOL    41
      P=PIVOT+1                                                         MATSOL    42
      III=N*PIVOT-N                                                     MATSOL    43
      IIPP=III+PIVOT                                                    MATSOL    44
      DO 8 I=P,N                                                        MATSOL    45
      IPII=I+III                                                        MATSOL    46
      T=-A(IPII)/A(IIPP)                                                MATSOL    47
      IF (T) 6,8,6                                                      MATSOL    48
    6 JJ=N*P-N                                                          MATSOL    49
      DO 7 J=P,N                                                        MATSOL    50
      II=I+JJ                                                           MATSOL    51
      JJPP=JJ+PIVOT                                                     MATSOL    52
      JJ=JJ+N                                                           MATSOL    53
    7 A(II)=A(II)+T*A(JJPP)                                             MATSOL    54
    8 CONTINUE                                                          MATSOL    55
    9 ERROR=0.0                                                         MATSOL    56
      GOTO 13                                                           MATSOL    57
   10 WRITE(6,11)N                                                      MATSOL    58
```

```
      ERROR=1.0                                                          MATSOL     59
   11 FORMAT      (40H0*****   SUBROUTINE MATSOL ERROR   *****,/,51H THE MATSOL     60
     1 SYSTEM OF EQUATIONS IS SINGULAR.  (NOTE -  N =,I4/)               MATSOL     61
      RETURN                                                             MATSOL     62
C  SOLVE THE SYSTEM BY BACK SUBSTITUTION.                                MATSOL     63
   13 L=N-1                                                              MATSOL     64
      DO 15 I=1,L                                                        MATSOL     65
      IP=I+1                                                             MATSOL     66
      II=N*I-N                                                           MATSOL     67
      IPII=I+II                                                          MATSOL     68
      YA=Y(I)/A(IPII)                                                    MATSOL     69
      DO 15 J=IP,N                                                       MATSOL     70
      JPII=J+II                                                          MATSOL     71
   15 Y(J)=Y(J)-A(JPII)*YA                                               MATSOL     72
      I=N                                                                MATSOL     73
      IN=I+N*I-N                                                         MATSOL     74
      Y(I)=Y(I)/A(IN)                                                    MATSOL     75
   16 IQ=I                                                               MATSOL     76
      I=I-1                                                              MATSOL     77
      IF(I) 19,19,17                                                     MATSOL     78
   17 T=Y(I)                                                             MATSOL     79
      INJ=I+N*IQ-N                                                       MATSOL     80
      DO 18 J=IQ,N                                                       MATSOL     81
      T=T-A(INJ)*Y(J)                                                    MATSOL     82
      INJ=INJ+N                                                          MATSOL     83
   18 CONTINUE                                                           MATSOL     84
      IN1=I+N*I-N                                                        MATSOL     85
      Y(I)=T/A(IN1)                                                      MATSOL     86
      GOTO 16                                                            MATSOL     87
   19 DO 20 I=1,N                                                        MATSOL     88
      J=IX(I)                                                            MATSOL     89
      J=NA(J)                                                            MATSOL     90
   20 V(J)=Y(I)                                                          MATSOL     91
      RETURN                                                             MATSOL     92
      END                                                                MATSOL     93
```

```
      SUBROUTINE CONT(HP,HPP,Z0,Z0P,Z1,Z1P,Z2)                          CONT       2
C  CALLED BY MRK.                                                       CONT       3
C  CONT COMPUTES THE COEFFICIENTS TO ESTIMATE THE TRUNCATION ERROR.      CONT       4
C  A FOURTH DEGREE POLYNOMIAL IS FITTED TO THE DATA, AND ITS            CONT       5
      UP=HP                                                             CONT       6
      UPP=HP+HPP                                                        CONT       7
      U0=UPP-UP                                                         CONT       8
      UPS=UP**2                                                         CONT       9
      UPPS=UPP**2                                                       CONT      10
      DET=UP*U0**2                                                      CONT      11
      Z0=((UP-U0-U0)/UPS-UP/UPPS )/DET                                  CONT      12
      Z0P=(U0**2/(UP*UPP))/DET                                          CONT      13
      Z1=((U0+U0-UP)/UPS )/DET                                          CONT      14
      Z1P=UD/(UP*DET)                                                   CONT      15
      Z2=UP/(UPPS*DET)                                                  CONT      16
      RETURN                                                            CONT      17
      END                                                               CONT      18
```

```
      SUBROUTINE PDTR(CDO,CO1,CO2,DU,H,HP,HPP)                          PDTR       2
C  CALLED BY MRK.                                                       PDTR       3
C  PDTR FINDS THE COEFFICIENTS FOR THE TWO STEP FORMULA                  PDTR       4
C  PREDICTING F AND R.                                                   PDTR       5
      CD2=(H*(H+HP))/(HPP*(HP+HPP))                                     PDTR       6
      CO1=(-H*(H+HP+HPP))/(HP*HPP)                                      PDTR       7
      COD=1.0-CO1-CD2                                                   PDTR       8
      CDD=1.0/CDO                                                       PDTR       9
      CO1=CD1*CDO                                                       PDTR      10
      CD2=CO2*COO                                                       PDTR      11
      RETURN                                                            PDTR      12
      END                                                               PDTR      13
```

```
      SUBROUTINE DER(Y,DERY)                                          DER        2
C  CALLED BY MRK.                                                     DER        3
C  DER IS USER SUPPLIED.                                              DER        4
C  IT COMPUTES THE REQUIRED DERIVATIVES AND JACOBIAN ELEMENTS.         DER        5
      DIMENSION Y(30),DERY(30)                                        DER        6
      COMMON/TAB2/W(30,30),KEY(30)                                    DER        7
      COMMON/TAB3/NDIM,IDIM                                           DER        8
      COMMON/TAB4/NITG,NDER,NJAC,NM1,NDM1                             DER        9
      Y1=Y(1)                                                         DER       10
      Y2=Y(2)                                                         DER       11
      Y3=Y(3)                                                         DER       12
      Y4=Y(4)                                                         DER       13
C  FIND THE TIME DERIVATIVES.                                         DER       14
      DERY(1)=-Y1+2.0                                                 DER       15
      DERY(2)=-10.*Y2+20.*Y1*Y1                                       DER       16
      DERY(3)=-40.*Y3+80.*(Y1*Y1+Y2*Y2)                               DER       17
      DERY(4)=-100.*Y4+200.*(Y1*Y1+Y2*Y2+Y3*Y3)                       DER       18
      NDER=NDER+1                                                     DER       19
      RETURN                                                          DER       20
      ENTRY JACOB                                                     DER       21
C  FIND THE DIAGONAL MODIFIED JACOBIAN ELEMENTS.                      DER       22
      W(1,1)=-Y1                                                      DER       23
      W(2,2)=-10.0*Y2                                                 DER       24
      W(3,3)=-40.0*Y3                                                 DER       25
      W(4,4)=-100.0*Y4                                                DER       26
      RETURN                                                          DER       27
      ENTRY JAC                                                       DER       28
C  FIND THE OFF DIAGONAL MODIFIED JACOBIAN ELEMENTS THAT ARE           DER       29
C  BEING USED IN THE CURRENT TIME STEP.                                DER       30
      DO 500 I=1,NDIM                                                 DER       31
      DO 500 J=1,NDIM                                                 DER       32
      IF(I.EQ.J)GO TO 500                                             DER       33
      W(I,J)=0.0                                                      DER       34
500   CONTINUE                                                        DER       35
      IF(KEY(2).LT.1)GOTO 3                                           DER       36
      W(2,1)=40.*Y1*Y1                                                DER       37
    3 IF(KEY(3).LT.1)GOTO 4                                           DER       38
      W(3,1)=160.*Y1*Y1                                               DER       39
      W(3,2)=160.*Y2*Y2                                               DER       40
    4 IF(KEY(4).LT.1)GOTO 5                                           DER       41
      W(4,1)=400.*Y1*Y1                                               DER       42
      W(4,2)=400.*Y2*Y2                                               DER       43
      W(4,3)=400.*Y3*Y3                                               DER       44
    5 CONTINUE                                                        DER       45
      NJAC=NJAC+1                                                     DER       46
      RETURN                                                          DER       47
      END                                                             DER       48
```

```
          S C O P E   2   L O A O   M A P                    LOADER VERSION 1.0   03/19/78  14.19.35.  PAGE   1

PROGRAM  WILL BE ENTEREO AT  KINT       (  150)               SCM LENGTH      11645          LCM LENGTH          0


          BLOCK         AOORESS     LENGTH       FILE

           /TAB1/          110          5        LGO
           /TAB3/          115          2        LGO
           /TAB4/          117          5        LGO
          K1NT             124        454        LGO
           /TAB2/          600       1642        LGO
          MRK             2442       2340        LGO
          FIX             5002        173        LGO
          MATSOL          5175        370        LGO
          CONT            5565         35        LGO
          POTR            5622         21        LGO
          OER             5643        141        LGO
           /STP.ENO/      6004          1     SL-FTNLIB
           /FCL.C./       6005         23     SL-FTNLIB
           /Q8.10./       6030        136     SL-FTNLIB
          QBNTRY=         6166          1     SL-FTNLIB
          COMIO=          6167         44     SL-FTNLIB
          FECMSK=         6233         41     SL-FTNLIB
          FLTOUT=         6274        315     SL-FTNLIB
          FMTAP=          6611        373     SL-FTNLIB
          FORSYS=         7204        533     SL-FTNLIB
          FORUTL=         7737         44     SL-FTNLIB
          GETF1T=        10003         43     SL-FTNLIB
          KOOER=         10046        467     SL-FTNLIB
          OUTC=          10535        171     SL-FTNLIB
          OUTCOM=        10726        204     SL-FTNLIB
          CLOCK=         11132         43     SL-FTNLIB
          ALOG           11175         77     SL-FTNLIB
          EXP            11274        100     SL-FTNLIB
          SQRT           11374         46     SL-FTNLIB
          SYSA10=        11442          1     SL-FTNLIB
          SYS=1ST        11443         62     SL-FTNLIB
          XT01*          11525         33     SL-FTNLIB
          XT01=          11560         10     SL-FTNLIB
          XTOY*          11570         55     SL-FTNLIB


 ERR CRIT =  1.0000E-08
 THE UPPER BOUNO FOR THE INITIAL STEP SIZE IS     .1000E-03

RUN TIME =  6.2000E-02

MRK CALLS =   302      OERIVATIVE CALLS =   608      JACOBEAN CALLS =      6

NO. OF MATRIX INVERSIONS =      6      AVERAGE OIMENSION =  2.0000E+00

  1.064686985169E+00  1.560027657738E+00  5.820783762433E+00  6.283142314819E+01

FINAL TIME =  6.6874E-02

    ERROR =  0.


 ERR CRIT =  1.0000E-06
```

```
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  2.0000E-02

MRK CALLS =    98      DERIVATIVE CALLS =   198      JACOBEAN CALLS =     7

NO. OF MATRIX INVERSIONS =     7      AVERAGE DIMENSION =  1.1429E+00

  1.064686985165E+00  1.560027643336E+00  5.820782077036E+00  6.283138999464E+01

FINAL TIME =  6.6874E-02

    ERROR =  1.6598E-05


 ERR CRIT =  1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  8.0000E-03

MRK CALLS =    36      DERIVATIVE CALLS =    75      JACOBEAN CALLS =     5

NO. OF MATRIX INVERSIONS =     5      AVERAGE DIMENSION =  1.0000E+00

  1.064686985040E+00  1.560027220819E+00  5.820729835103E+00  6.283018040171E+01

FINAL TIME =  6.6874E-02

    ERROR =  6.2196E-04


 ERR CRIT =  1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  4.0000E-03

MRK CALLS =    19      DERIVATIVE CALLS =    40      JACOBEAN CALLS =     1

NO. OF MATRIX INVERSIONS =     1      AVERAGE DIMENSION =  1.0000E+00

  1.064686983562E+00  1.560022085858E+00  5.820054963861E+00  6.281755530273E+01

FINAL TIME =  6.6874E-02

    ERROR =  6.9435E-03


 ERR CRIT =  1.0000E-08
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  1.5200E-01

MRK CALLS =   710      DERIVATIVE CALLS =  1433      JACOBEAN CALLS =   117

NO. OF MATRIX INVERSIONS =   117      AVERAGE DIMENSION =  1.0598E+00

  1.360592680788E+00  3.331503023687E+00  2.459212912655E+01  1.184760458930E+03

FINAL TIME =  4.4721E-01
```

```
    ERROR =  0.


 ERR CRIT =  1.0000E-06
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  5.4000E-02

MRK CALLS =   227      DERIVATIVE CALLS =   481      JACOBEAN CALLS =   107

NO. OF MATRIX INVERSIONS =   107      AVERAGE DIMENSION =  1.0187E+00

  1.360592679192E+00  3.331502847640E+00  2.459212622392E+01  1.184760155095E+03

FINAL TIME =  4.4721E-01

    ERROR =  1.5192E-04


 ERR CRIT =  1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  1.9000E-02

MRK CALLS =    76      DERIVATIVE CALLS =   161      JACOBEAN CALLS =    44

NO. OF MATRIX INVERSIONS =    44      AVERAGE DIMENSION =  1.1818E+00

  1.360592638623E+00  3.331498164931E+00  2.459204996244E+01  1.184752221933E+03

FINAL TIME =  4.4721E-01

    ERROR =  4.1187E-03


 ERR CRIT =  1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  8.0000E-03

MRK CALLS =    35      DERIVATIVE CALLS =    74      JACOBEAN CALLS =    16

NO. OF MATRIX INVERSIONS =    16      AVERAGE DIMENSION =  1.0000E+00

  1.360592212851E+00  3.331444363491E+00  2.459120140067E+01  1.184662824373E+03

FINAL TIME =  4.4721E-01

    ERROR =  4.8819E-02


 ERR CRIT =  1.0000E-08
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  5.8300E-01

MRK CALLS =  2470      DERIVATIVE CALLS =  5090      JACOBEAN CALLS =  1271

NO. OF MATRIX INVERSIONS =  1271      AVERAGE DIMENSION =  1.0110E+00
```

```
  1.949747629598E+00  7.559625514936E+00  1.215530823603E+02  2.960426798111E+04

FINAL TIME =  2.9907E+00

    ERROR =  0.


 ERR CRIT =  1.0000E-06
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  2.1100E-01

MRK CALLS =   792      DERIVATIVE CALLS =  1738      JACOBEAN CALLS =   645

NO. OF MATRIX INVERSIONS =   645      AVERAGE DIMENSION =  1.0186E+00

  1.949747627407E+00  7.559625496519E+00  1.215530817893E+02  2.960426770378E+04

FINAL TIME =  2.9907E+00

    ERROR =  1.3866E-04


 ERR CRIT =  1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  6.6000E-02

MRK CALLS =   243      DERIVATIVE CALLS =   532      JACOBEAN CALLS =   211

NO. OF MATRIX INVERSIONS =   211      AVERAGE DIMENSION =  1.1137E+00

  1.949747551756E+00  7.559624851115E+00  1.215530613720E+02  2.960425772858E+04

FINAL TIME =  2.9907E+00

    ERROR =  5.1263E-03


 ERR CRIT =  1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  2.4000E-02

MRK CALLS =    87      DERIVATIVE CALLS =   188      JACOBEAN CALLS =    69

NO. OF MATRIX INVERSIONS =    69      AVERAGE DIMENSION =  1.347ME+00

  1.949745357133E+00  7.559606182033E+00  1.215524718396E+02  2.960397049424E+04

FINAL TIME =  2.9907E+00

    ERROR =  1.4874E-01


 ERR CRIT =  1.0000E-08
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03
```

```
RUN TIME =  1.1310E+00

MRK CALLS =  4517      DERIVATIVE CALLS =  9414      JACOBEAN CALLS =  3282

NO. OF MATRIX INVERSIONS =  3282      AVERAGE DIMENSION =  1.0545E+00

  1.999999997939E+00  7.999999981676E+00  1.359999993817E+02  3.712799965962E+04

FINAL TIME =  2.0000E+01

    ERROR =  0.


 ERR CRIT =  1.0000E-06
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  3.7500E-01

MRK CALLS =  1382      DERIVATIVE CALLS =  2979      JACOBEAN CALLS =  1248

NO. OF MATRIX INVERSIONS =  1248      AVERAGE DIMENSION =  1.1482E+00

  1.999999997927E+00  7.999999981572E+00  1.359999993782E+02  3.712799965770E+04

FINAL TIME =  2.0000E+01

    ERROR =  9.6008E-07


 ERR CRIT =  1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  1.4300E-01

MRK CALLS =   480      DERIVATIVE CALLS =  1115      JACOBEAN CALLS =   467

NO. OF MATRIX INVERSIONS =   467      AVERAGE DIMENSION =  1.2934E+00

  1.999999997843E+00  7.999999980828E+00  1.359999987363E+02  3.712799964636E+04

FINAL TIME =  2.0000E+01

    ERROR =  6.6390E-06


 ERR CRIT =  1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS    .1000E-03

RUN TIME =  5.7000E-02

MRK CALLS =   183      DERIVATIVE CALLS =   428      JACOBEAN CALLS =   165

NO. OF MATRIX INVERSIONS =   165      AVERAGE DIMENSION =  1.7273E+00

  1.999999996770E+00  7.999999971291E+00  1.359999990312E+02  3.712799946670E+04

FINAL TIME =  2.0000E+01

    ERROR =  9.6462E-05

  **********    TERRYU2   0001006 LINES PRINTED
```

DISTRIBUTION LIST

| No. of Copies | Organization |
|---|---|
| 12 | Commander<br>Defense Technical Info Center<br>ATTN: DDC-DDA<br>Cameron Station<br>Alexandria, VA 22314 |
| 1 | Commander<br>US Army Materiel Development and Readiness Command<br>ATTN: DRCDMD-ST<br>5001 Eisenhower Avenue<br>Alexandria, VA 22333 |
| 2 | Commander<br>US Army Armament Research and Development Command<br>ATTN: DRDAR-TSS<br>Dover, NJ 07801 |
| 1 | Commander<br>US Army Armament Materiel Readiness Command<br>ATTN: DRSAR-LEP-L, Tech Lib<br>Rock Island, IL 61299 |
| 1 | Director<br>US Army ARRADCOM<br>Benet Weapons Laboratory<br>ATTN: DRDAR-LCB-TL<br>Watervliet, NY 12189 |
| 1 | Commander<br>US Army Aviation Research and Development Command<br>ATTN: DRSAV-E<br>P.O. Box 209<br>St. Louis, MO 63166 |
| 1 | Director<br>US Army Air Mobility Research and Development Laboratory<br>Ames Research Center<br>Moffett Field, CA 94035 |
| 1 | Commander<br>US Army Communications Rsch and Development Command<br>ATTN: DRDCO-PPA-SA<br>Fort Monmouth, NJ 07703 |
| 1 | Commander<br>US Army Electronics Research and Development Command<br>Technical Support Activity<br>ATTN: DELSD-L<br>Fort Monmouth, NJ 07703 |
| 1 | Director<br>US Army Electronics Command<br>Atmospheric Sciences Laboratory<br>ATTN: M.G. Heaps<br>White Sands Missile Range, NM 88002 |
| 2 | Commander<br>US Army Missile Command<br>ATTN: DRDMI-R<br>DRDMI-YDL<br>Redstone Arsenal, AL 35809 |
| 1 | Commander<br>US Army Tank Automotive Research & Development Command<br>ATTN: DRDTA-UL<br>Warren, MI 48090 |
| 3 | Commander<br>US Army Research Office<br>ATTN: J. Chandra<br>D. Squire<br>R. Lontz<br>P.O. Box 12211<br>Research Triangle Park, NC 27709 |

DISTRIBUTION LIST

| No.of Copies | Organization |
|---|---|
| 1 | Director<br>US Army TRADOC Systems Analysis Activity<br>ATTN: ATAA-SL, Tech Lib<br>White Sands Missile Range<br>NM 88002 |
| 5 | Commander<br>Naval Research Laboratory<br>ATTN: W. Ali<br>D. Strobel<br>J. Brown, Code 7700<br>E. Oran, Code 6020<br>Tech Lib, Code 2020<br>Washington, DC 20375 |
| 3 | AFGL (LKB, J. Paulson,<br>K. Champion,<br>W. Swider)<br>Hanscom AFB, MA 01731 |
| 2 | Director<br>National Bureau of Standards<br>US Department of Commerce<br>ATTN: R.F. Hampson<br>R. Kraft, Div 205.01<br>Washington, DC 20234 |
| 3 | Lockheed Palo Alto Rsch Lab<br>ATTN: J. Reagan<br>R. Sears<br>Tech Info Ctr<br>3521 Hanover Street<br>Palo Alto, CA 94304 |
| 1 | Director<br>National Aeronautics and Space Administration<br>Goddard Space Flight Center<br>ATTN: R. Goldberg<br>(Code 912)<br>Greenbelt, MD 20771 |

| No. of Copies | Organization |
|---|---|
| 1 | Bell Laboratories<br>ATTN: D. Edelson<br>Murray Hill, NJ 07974 |
| 1 | Boeing Computer Services, Inc.<br>Energy Technology Applications<br>ATTN: K.W. Neves<br>P.O. Box 24346<br>Seattle, WA 98124 |
| 1 | General Electric Company<br>Valley Forge Space Technology Center<br>ATTN: M. Bortner<br>P.O. Box 8555<br>Philadelphia, PA 19101 |
| 1 | General Electric Company<br>Flight Propulsion Division<br>ATTN: Tech Lib<br>Cincinnati, OH 45215 |
| 1 | Hercules Incorporated<br>Alleghany Ballistic Lab<br>ATTN: Tech Lib<br>Cumberland, MD 21501 |
| 1 | Mission Research Corporation<br>ATTN: M. Scheibe<br>735 State Street<br>P.O. Drawer 719<br>Santa Barbara, CA 93102 |
| 1 | Pacific-Sierra Research Corp.<br>ATTN: E. Fields<br>1456 Cloverfield Blvd<br>Santa Monica, CA 90404 |
| 1 | Paul Gough Associates, Inc.<br>ATTN: P.S. Gough<br>P.O. Box 1614<br>Portsmouth, NH 03801 |
| 1 | Physics International Company<br>2700 Merced Street<br>Leandro, CA 04577 |

DISTRIBUTION LIST

| No. of Copies | Organization |
|---|---|
| 1 | R&D Associates<br>ATTN: F. Gilmore<br>P.O. Box 9695<br>Marina Del Rey, CA 90291 |
| 1 | Science Applications, Inc.<br>ATTN: R.B. Edelman<br>23146 Cumorah Crest<br>Woodland Hills, CA 91364 |
| 3 | SRI International<br>ATTN: J. Peterson<br>D. Golden<br>Tech Lib<br>333 Ravenswood Avenue<br>Menlo Park, CA 94025 |
| 2 | Thiokol Corporation<br>Huntsville Division<br>ATTN: D. Flanigan<br>Tech Lib<br>Huntsville, AL 35807 |
| 1 | Battelle Memorial Institute<br>ATTN: Tech Lib<br>505 King Avenue<br>Columbus, OH 43201 |
| 1 | Brigham Young University<br>Dept of Chemical Engineering<br>ATTN: R. Coates<br>Provo, UT 84601 |
| 1 | Georgia Institute of Tech<br>School Of Aerospace Engrg<br>ATTN: B.T. Zinn<br>Atlanta, GA 30332 |
| 1 | Institute of Gas Technology<br>ATTN: D. Gidaspow<br>3424 S. State Street<br>Chicago, IL 60616 |
| 2 | Johns Hopkins University/APL<br>Chemical Propulsion Info Agency<br>ATTN: T. Christian<br>Johns Hopkins Road<br>Laurel, MD 20810 |
| 1 | Massachusetts Institute of Technology<br>Dept of Mechanical Engineering<br>ATTN: T. Toong<br>Cambridge, MA 02139 |
| 1 | Director<br>Graduate Center of Applied Science<br>New York University<br>ATTN: M. Summerfield<br>26-36 Stuyvesant<br>New York, NY 10003 |
| 1 | Oregon State University<br>Department of Mathematics<br>ATTN: D.C. Jespersen<br>Corvallis, OR 97331 |
| 1 | Pennsylvania State University<br>Dept of Mechanical Engineering<br>ATTN: K. Kuo<br>University Park, PA 16801 |
| 2 | Princeton University<br>Forrestal Campus Library<br>ATTN: L. Caveny<br>Tech Lib<br>P.O. Box 710<br>Princeton, NJ 08540 |
| 1 | Purdue University<br>Indianapolis Regional Campus<br>ATTN: J.M. Gersting<br>1201 East 38th Street<br>Indianapolis, IN 46205 |

DISTRIBUTION LIST

| No. of Copies | Organization |
|---|---|
| 2 | Purdue Unviersity<br>School of Mechanical Engrg<br>ATTN: J. Osborn<br>S.N.B. Murthy<br>TSPC Chaffee Hall<br>West Lafayette, IN 47906 |
| 1 | University of California<br>Electrical Engineering and<br>Computer Science Dept<br>ATTN: W. Kahan<br>567 Evans Hall<br>Berkeley, CA 94720 |
| 1 | University of California<br>ATTN: A.C. Hindmarsh, L-310<br>Lawrence Livermore Laboratory<br>P.O. Box 808<br>Livermore, CA 94550 |
| 1 | University of California,<br>San Diego<br>AMES Department<br>ATTN: F. Williams<br>P.O. Box 109<br>La Jolla, CA 92037 |
| 1 | University of Illinois<br>Dept of Aeronautical Engrg<br>ATTN: H. Krier<br>Transportation Bldg, Rm 105<br>Urbana, IL 61801 |
| 1 | University of Illinois<br>Dept of Electrical Engineering<br>ATTN: C. Sechrist, Jr.<br>Urbana-Champaign Campus<br>Urbana, IL 61801 |
| 1 | University of Wisconsin<br>Dept of Chemical Engineering<br>ATTN: J.R. Bowen<br>1415 Johnson Drive<br>Madison, WI 53706 |
| 2 | University of Wisconsin - Madison<br>Mathematics Research Center<br>ATTN: J. Nohel, Director<br>B. Noble<br>610 Walnut Street<br>Madison, WI 53706 |
| 1 | Utah State University<br>Center for Research in Aeronomy<br>ATTN: L. Megill<br>Logan, UT 84321 |
| 1 | Wayne State University<br>Department of Engineering<br>ATTN: R. Kummler<br>Detroit, MI 48202 |

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: DRXSY-D
DRXSY-MP, H. Cohen

Cdr, USATECOM
ATTN: DRSTE-TO-F

Dir, Wpns Sys Concepts Team,
Bldg E3516, EA
ATTN: DRDAR-ACW

## USER EVALUATION OF REPORT

Please take a few minutes to answer the questions below; tear out this sheet and return it to Director, US Army Ballistic Research Laboratory, ARRADCOM, ATTN: DRDAR-TSB, Aberdeen Proving Ground, Maryland 21005. Your comments will provide us with information for improving future reports.

1. BRL Report Number _____

2. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

_____

_____

_____

3. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.) _____

_____

_____

4. Has the information in this report led to any quantitative savings as far as man-hours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

_____

_____

5. General Comments (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.) _____

_____

_____

_____

6. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____

Telephone Number: _____

Organization Address: _____

_____

_____